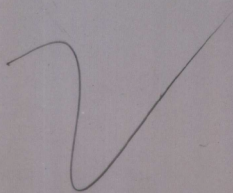




华夏英才基金学术文库

黄席樾 张著洪 何传江 著
胡小兵 马笑潇

现代智能算法 理论及应用



科学出版社

www.sciencep.com

(TP-2922.0101)

ISBN 7-03-015332-4



9 787030 153326 >

ISBN 7-03-015332-4
定 价: 49.00 元



华夏英才基金学术文库

现代智能算法理论及应用

黄席樾 张著洪 何传江 著
胡小兵 马笑潇

北 京

内 容 简 介

本书主要论述了智能算法中的免疫算法、分形编码算法、蚁群优化算法和支持向量机等问题。首先针对几类不同类型的一般性最优化问题,建立相应的基于免疫的算法,并进行理论和应用研究;其次介绍分形编码算法的理论基础及实现,探讨该算法的改进和应用;然后介绍蚁群优化算法的基本原理及并行实现,探讨其在工程问题中的应用;最后介绍支持向量机的统计学习理论基础,研究其在数据分类、故障诊断及故障预测中的应用。

全书取材新颖,覆盖面较广,深入浅出,注重算法的理论依据、应用思路及应用效果,体现了国内外在这方面研究的最新研究进展。本书可作为从事计算智能研究的科研人员及工程技术人员的参考书,也可供高等工科院校自动控制、计算机、通讯及导航与制导等相关专业的教师及研究生阅读。

图书在版编目(CIP)数据

现代智能算法理论及应用 / 黄席樾等著. —北京: 科学出版社, 2005
(华夏英才基金学术文库)

ISBN 7-03-015332-4

I. 现… II. 黄… III. 人工智能-算法理论 IV. TP183

中国版本图书馆 CIP 数据核字(2005)第 029432 号

责任编辑: 李 锋 范庆奎 / 责任校对: 宋玲玲

责任印制: 钱玉芬 / 封面设计: 王 浩

科 学 出 版 社 出 版

北京东黄城根北街16号

邮政编码: 100717

<http://www.sciencep.com>

双青印刷厂印刷

科学出版社发行 各地新华书店经销

*

2005年4月第 一 版 开本: 85(720×1000)

2005年4月第一次印刷 印张: 27 1/2

印数: 1—4 000 字数: 544 000

定价: 49.00 元

(如有印装质量问题, 我社负责调换〈环伟〉)

前 言

以人工神经网络、演化计算等为代表的智能算法在工程领域的成功应用,激励人们从更广泛的生物或自然现象寻求启发以构造新的智能算法,来解决工程中广泛存在的复杂问题.这种以生物智能或自然现象为基础的随机搜索算法具有比数学规划方法更大的优越性,主要表现在:(1)具有一般性且易于应用;(2)搜索最优解的速度快且易于获得满意结果.以上优点使这类算法在工程问题上具有更广泛的应用前景,从而吸引了更多学者对其进行研究,使得现代智能算法正在成为人工智能领域一个新的研究热点.

本书是作者多年来对人工免疫算法、分形算法、蚁群优化算法和支持向量机的理论及应用进行了一系列研究并在所得成果的基础上总结而成的一本专著,内容反映了这四大分支的国内外最新研究动态.全书包括四部分,共十六章,其中第一章绪论是综述;第二章至第五章研究人工免疫系统的智能优化及免疫网络;第六章至第八章探讨分形算法理论及应用;第九章至第十三章研究蚁群优化算法理论及应用;第十四章至第十六章探讨支持向量机在故障诊断中的应用.各章内容安排如下:

第二章介绍免疫学的基本概念及原理,给出各原理中包含的运行机制之间的结构关系框架;对人工免疫系统理论及应用的发展历程及研究现状作了简要回顾,综述基于免疫机理的智能优化的发展阶段及免疫网络算法的主要结果.

第三章探讨一般性免疫算法的思路及与体液免疫应答的对应关系;对齐次及非齐次免疫算法的收敛性及收敛速度估计进行理论探讨;对免疫算法稳定性展开研究,探讨这类问题的自身理论体系;从实验角度研究免疫算法的鲁棒性,并对齐次及非齐次免疫算法进行理论分析和进行大量性能测试、比较.

第四章介绍基于免疫函数优化算法的研究存在的问题;给出小生境免疫算法及动态规模免疫算法,处理非约束函数优化问题,并对这两种算法进行性能检测、比较;提出约束优化免疫算法处理约束优化问题,并对其性能检测、比较;对小生境免疫算法、动态规模免疫算法及约束优化免疫算法的收敛性进行理论研究,并给出算法的实际应用.

第五章介绍智能算法解决多目标优化问题的发展现状,以及免疫网络算法处理数据分类问题的发展;提出多目标优化免疫算法,解决非约束多目标优化问题,进行收敛性论证及性能测试和比较;提出约束多目标优化免疫算法解决约束多目标优化问题;提出模糊免疫网络聚类算法处理分类问题,并用随机产生的样本组进行检测.

第六章介绍分形编码的理论基础,包括分形的概念、迭代函数系统、压缩映射原理和拼贴定理.

第七章从迭代函数系统的观点介绍分形编码的基本原理,并从 VQ 的观点介绍该算法及实现。

第八章介绍作者的部分研究成果,特别是发表在 IEE 刊物上的结果。

第九章综述蚁群优化算法的发展、应用及国内外研究现状;通过“双桥实验”分析了蚁群优化算法的基本原理,并由此导出蚁群优化算法的基本数学模型;将蚁群优化算法与其他算法进行比较。

第十章介绍简单蚁群优化算法及其特性,通过该算法分析了真实蚂蚁与人工蚂蚁之间的异同;介绍蚁群优化元启发式;分析蚁群优化算法中解的隐式评估、显式评估、信息素的留存和蚂蚁数等对算法的影响。

第十一章介绍蚂蚁系统的模型及实现,并对该算法中的参数设置、蚂蚁数的选择、停滞现象、信息素的分布、蚂蚁协同作用和蚂蚁的初始分布等对算法的影响进行讨论;介绍几种改进的蚁群优化算法,并对各种算法的性能进行对比。

第十二章介绍蚁群优化算法并行实现的主要途径及其研究现状;对比分析两种典型的蚁群优化算法的并行实现——同步并行实现和部分异步并行实现;介绍带聚类处理的并行蚁群优化算法。

第十三章介绍蚁群优化算法在各个领域的应用,包括 K-TSP、二次分配问题、作业调度问题、网络路由问题、0-1 背包问题、迷宫问题和机器人路径规划问题。

第十四章研究小样本情况的统计学习规律,探讨支持向量机的统计学理论基础,阐述从经验风险最小化到结构风险最小化的统计学习变革历程,给出 SVM 学习机的推导算法;利用一个可视化方便的 2 维两类分类的实例给出在以径向基、多项式和 Sigmoid 函数为核函数情况下的分类结果。

第十五章研究基于 SVM 的多类模式识别算法,探讨支持向量机用于故障诊断的关键问题;以柴油机振动信号的故障诊断为例,利用基于 SVM 的二叉树多类分类方法判断柴油机故障检测和故障原因,分析 SVM 理论中有关参数的特性和选择依据。

第十六章研究 SVM 回归算法的基本实现方法,给出运用 SVM 机器学习方法进行故障趋势预测的方法,通过对“Tennessee Eastman”工厂的实际数据进行仿真,分析用 SVM 方法进行故障趋势预测的优越性。

本书由黄席樾教授执笔撰写第一、二、三、六、九、十和十四章;由张著洪博士执笔撰写第四、五章;由何传江博士执笔撰写第七、八章;由胡小兵博士执笔撰写第十一、十二和十三章;由马笑潇博士执笔撰写第十五、十六章。

全书内容取材新颖,覆盖面较广,深入浅出,注重理论联系实际,可作为高等院校相关专业研究生的专业参考书,也可作为相关教师及工程技术人员参考书籍。

由于作者水平有限,缺点和错误在所难免,敬请广大同行、读者批评指正。

作者

2004 年 12 月

目 录

前言

第一章 绪论	1
第一节 人工免疫系统概述	4
第二节 分形编码概述	6
第三节 蚁群优化概述	8
第四节 支持向量机概述	10

第一部分 免疫优化及免疫网络算法理论和应用

第二章 免疫学基本理论及人工免疫系统概论	15
第一节 免疫学基本概念及原理	15
第二节 人工免疫系统概述	22
第三节 本篇研究的主要内容及意义	30
第四节 最优化问题及分类	32
第五节 测试问题及性质分析	32
第六节 本章小结	36
参考文献	36

第三章 免疫算法理论及应用	41
第一节 引言	41
第二节 免疫算法的概念及工作原理	43
第三节 免疫算子及相关概念	45
第四节 突变规则	48
第五节 免疫算法描述	49
第六节 算法收敛性概念	51
第七节 免疫算子性质及齐次免疫算法收敛性	53
第八节 非齐次免疫算法收敛性	57
第九节 免疫算法收敛速度分析	60
第十节 免疫算法稳定性理论	63
第十一节 免疫算法的计算复杂度及鲁棒性分析	73
第十二节 齐次及非齐次免疫算法理论比较分析	78

第十三节 免疫算法的性能测试	81
第十四节 应用举例	86
第十五节 本章小结	87
参考文献	88
第四章 形态空间上免疫算法及收敛性理论	89
第一节 引言	89
第二节 小生境免疫算法	90
第三节 动态规模免疫算法	94
第四节 约束优化免疫算法	97
第五节 模糊控制免疫算法	104
第六节 形态空间上免疫算法的收敛性	111
第七节 应用举例	116
第八节 本章小结	123
参考文献	123
第五章 多目标优化免疫算法及免疫网络算法	125
第一节 引言	125
第二节 预备知识	126
第三节 非约束条件下多目标优化免疫算法	127
第四节 约束多目标优化免疫算法	138
第五节 模糊免疫网络分类算法	147
第六节 本章小结	153
参考文献	153
第二部分 图像编码的分形算法	
第六章 分形编码的数学基础	157
第一节 引言	157
第二节 度量空间	157
第三节 分形	168
第四节 迭代函数系统	175
第五节 本章小结	184
参考文献	184

第七章 基本分形编码算法	187
第一节 引言	187
第二节 矢量量化与分形编码	189
第三节 迭代函数系统正问题与自然图形模拟	197
第四节 迭代函数系统逆问题与图像编码	202
第五节 分形编码算法的基本原理与实现	204
第六节 本章小结	222
参考文献	222
第八章 分形编码的改进算法	225
第一节 引言	225
第二节 图像分割	227
第三节 虚拟码本构成	233
第四节 亮度变换类型	235
第五节 变换参数的量化	237
第六节 分形解码	240
第七节 最优分形编码	252
第八节 快速分形编码	255
第九节 混合分形编码	270
第十节 本章小结	275
参考文献	276
第三部分 蚁群优化算法理论及其应用	
第九章 蚁群优化算法概述	283
第一节 引言	283
第二节 蚁群优化原理及算法描述	284
第三节 蚁群优化的特点	289
第四节 蚁群优化与其他算法的关系	290
第五节 蚁群优化的研究现状	291
第六节 本章小结	294
参考文献	294
第十章 蚁群优化元启发式及其收敛性	301
第一节 引言	301

第二节 蚁群优化元启发式	301
第三节 蚁群优化的收敛性	307
第四节 本章小结	317
参考文献	317
第十一章 基本蚁群优化算法及其改进算法	319
第一节 引言	319
第二节 蚂蚁系统及其属性	319
第三节 改进的蚁群优化算法	326
第四节 一种新的自适应蚁群算法	332
第五节 基于混合行为的蚁群算法	335
第六节 本章小结	340
参考文献	341
第十二章 蚁群优化的并行实现	343
第一节 蚁群优化的并行实现概述	343
第二节 蚂蚁系统的同步并行实现和部分异步并行实现	344
第三节 SPI 与 PAPI 的对比实验	346
第四节 对一类带聚类特征 TSP 的并行蚁群算法求解	348
第五节 本章小结	356
参考文献	356
第十三章 蚁群优化算法的应用	357
第一节 概述	357
第二节 蚁群优化算法与 K-TSP	357
第三节 蚁群优化与二次分配问题	361
第四节 蚁群优化算法与车间作业调度问题	368
第五节 蚁群优化算法与网络路由问题	370
第六节 蚁群算法与 0-1 背包问题	372
第七节 蚁群优化算法与三维空间机器人路径规划	378
第八节 本章小结	382
参考文献	383

第四部分 小样本统计学习理论与支持向量机

第十四章 小样本统计学习的基本理论	387
第一节 引言	387
第二节 基于 SLT 的机器学习理论的基本观点	387
第三节 支持向量机算法	394
第四节 算例	399
第五节 本章小结	401
参考文献	401
第十五章 基于 SVM 的多类分类算法及其在故障诊断中的应用	403
第一节 引言	403
第二节 基于二叉树的多级 SVM 分类器	403
第三节 SVM 用于故障诊断的一般步骤	406
第四节 基于 SVM 的柴油机故障诊断	408
第五节 本章小结	416
参考文献	416
第十六章 基于支持向量机的函数回归的方法	418
第一节 常用的损失函数的定义	418
第二节 函数回归的 SVM 方法	419
第三节 基于 SVM 的故障趋势预测研究	421
第四节 本章小结	426
参考文献	426
中英文词汇对照	428

第一章 绪 论

智能算法是一种借鉴和利用自然界中自然现象或生物体的各种原理和机理而开发的并具有自适应环境能力的计算方法,衡量智能算法的智能程度高低的关键在于其处理实际对象时所表现出的学习能力的强弱.智能算法的发展已有较悠久的历史,早期发展起来的符号主义、联结主义、进化计算、模拟退火算法作为经典智能方法的主要研究学派,至今仍在计算智能领域占据着重要位置,并已取得极为丰硕的理论及应用成果.随着历史的变革和时代的变迁,智能算法的研究经历了漫长的发展过程,从早期的经典智能算法发展到现代智能算法,人们在不断探索新智能方法的同时,对经典智能方法进行了一系列反思.符号机制体系的建立,以知识表达为基础,通过推理求解的机器证明的成功实现,开创了计算机进行几何证明的革命时代;基于联结机制的人工神经网络简单地模拟人类大脑的学习功能,对用网络模型处理工程问题作出了巨大贡献,但是局部极小及搜索效率低是其主要不足;进化计算的发展,使得经典计算智能的研究再度掀起,致使智能算法成为当今研究的热点,并已发展为一种多学科、多智能交叉融合、渗透的信息与计算研究领域.经典智能算法与来自生命科学中其他生物理论的结合,使得这类算法有了较大进展,如遗传算法与生物免疫或模糊逻辑的结合形成了免疫遗传算法或模糊遗传算法,神经网络与免疫网络的结合形成了免疫神经网络.现代智能算法在经典智能算法的理论及应用基础上,已逐步发展出许多较有潜力的研究分支,如人工免疫系统理论及应用、分形算法理论及应用、蚁群优化算法理论及应用、支持向量机等等.这些表明智能算法已朝着多极化发展,生物智能的应用越来越广,研究学派越来越多,研究气氛越来越活跃,比如智能优化算法作为智能算法的重要研究内容,已呈现较多的新智能工具,如免疫遗传算法、免疫算法、混沌免疫算法、蚁群优化算法、噪声方法、变邻域搜索、巢分区方法、思维进化算法、混合优化算法等等.这些算法的出现显示了模拟或借鉴生物智能,开发具有较高智能的应用工具并对其进行理论与应用研究具有重要理论和现实意义.

目前智能算法的研究呈现出三大趋势:一是对经典智能算法的改进和广泛应用,以及对其理论的深入、广泛研究;二是现代智能算法的发展,即开发新的智能工具,拓宽其应用领域,并对其寻求理论基础;三是经典智能算法与现代智能算法的结合建立混合智能算法.至今新的智能算法不断涌现,涉及的应用领域不断增多,如最优化、模式识别、智能控制、计算机安全、计算机网络、投资组合等等,因此开发新智能工具处理工程问题便成为现代智能算法研究的首要任务,也是智能算法研究的热点.

为了使读者能更清楚地了解智能优化算法发展现状和趋势,以下首先对进化算法、模拟退火算法、处理优化问题的神经网络的发展现状、存在的问题以及进一步研究态势作概要性介绍,然后对本书的各研究分支的研究现状及本书研究内容作概述介绍。

智能算法是建立在生物智能或物理现象基础上的随机搜索算法,由于其自身作为启发式随机算法,具有比数学规划方法更优越的特性,因此这类方法的研究也是最优化领域研究的重点。其优点是:(1)具有一般性及易于应用;(2)搜索最优解的速度快且易于获得满意的结果。目前智能优化方法较多,许多方法存在着不同程度的相似性,如噪声方法、变邻域搜索、巢分区方法均属于邻域搜索算法,存在着与模拟退火算法所具有的许多共同特征。最为广泛被采用并具有代表性的智能优化方法是模拟退火算法(SA)、进化算法(EA)和Hopfield网络。以下仅对这三种研究学派的研究状况作简单回顾。

进化算法包括20世纪70年代中期美国的Holland提出的遗传算法GA、60年代德国的Rechenberg及Schwefel提出的进化策略ES及60年代美国的Fogel等教授提出的进化规划EP,而GA又是这三种算法的代表。由于这三种算法的算子具有可统一性,因此将此三种算法统称为进化算法EA。目前关于EA的一般性理论研究,德国的Rudolph获得了初步结果,研究范畴主要集中在算法的收敛性,但要求的条件较强,即算法的状态转移矩阵正定,这无疑导致应用受到极大限制,但是这项研究成果标志着一般框架的进化算法的理论探讨迈出了坚实的一步。

EA是模拟生物自然进化机制的一种启发式随机搜索算法,这种算法是以个体构成的群体为状态并具有开采和探测能力的群体学习过程,其由基于群体的选择、交叉及突变三种算子组成。选择操作反映了物种进化的“适者生存,优胜劣汰”思想,交叉是自然演化的主要机制,其反映了个体之间的信息传递和信息交换关系,突变使得群体搜索具有遍历性。选择、交叉、突变分别反映了物种进化过程的特定运行机制,这三种算子在GA、ES及EP中的重要程度有所不同。GA以选择和交叉为主产生多样和高适应度的个体,突变仅起到微调群体多样性的作用;ES及EP主要通过突变产生多样的个体,通过选择方式获得好个体;ES与EP的主要区别在于:ES的选择为确定性选择且含有交叉操作,而EP无交叉操作及选择为K联赛选择。GA的突出优点是结构相对固定、计算速度快、全局搜索能力强、并行性高、鲁棒性强及不依赖于问题的特征信息等,这些优点是导致大量计算智能研究者及工程人员对其产生浓厚兴趣的根本原因。无论是这种算法的理论研究还是应用技术开发,都在计算智能中占据了重要位置。

关于算法结构,GA是以多个个体构成当前状态,搜索过程是个体群进化,最终获得的状态是由多个优良个体组成。但GA维持群体多样性及局部搜索能力较弱,搜索性能与初始群体的分布密切相关,已有的理论结果已表明基本遗传算法是不收敛的,其主要原因在于尽管GA具有遍历性,但是算法搜索过程中所获得

的较好个体的维持性能较差及群体多样性不足, 导致任何时刻获得的好个体随时可能消失及易于出现早熟现象, 因而 GA 不能被保证收敛。由于 GA 具有的可能不收敛特性阻碍了其应用和理论的进一步深入研究, 这导致人们去寻找一些新的方法改善 GA 的搜索性能, 因而出现了基于 GA 的种类繁多的算法, 如模拟退火遗传算法、并行遗传算法、模糊遗传算法等。其次遗传算法处理约束函数优化问题的研究也取得了较大进展, 这些算法的出现有助于开发更有效的算法处理复杂约束优化问题; 另外以 GA 为基础的多目标优化进化算法作为进化算法的重要研究方向, 在近来已取得丰硕成果, 并在许多有关多目标优化领域获得广泛应用并引起同行研究人员的足够重视。在此研究领域里, Zitzler 提出的算法是目前公认的最好算法。这些算法不仅增强了 GA 的局部搜索能力和最优解维持能力, 而且使算法的搜索速度及处理复杂优化问题的能力极大提高, 特别是并行和模糊遗传算法及基于 GA 的多目标进化算法是计算智能领域里值得研究的、富有实际意义的研究方向, 同时也是目前 GA 的主要发展趋势。但当 GA 结构复杂程度加大时, 无疑增加了其理论研究的难度。在国内外, 对 GA 已有较系统的理论研究, 如西安交大的张文修教授等对 GA 的遗传算子的几何性质作了较全面探讨, 同时对建立在 GA 上的几类较为简单算法的收敛性作了深入研究, 提供了对智能优化方法进行理论分析的新思路, 获得了极为有价值的理论结果。为了改善 GA 的性能, 通常可通过对 GA 的结构复杂化或给予附加的特定的算子, 这当然提高了 GA 的搜索性能, 但也带来了负面影响, 即算法的通用性和实用性有所降低, 同时计算复杂度及理论研究的难度大幅度提高。

ES 是基于实数编码的保优算法, 其主要的算子是确定性的选择操作及高斯变异策略, 突出的优点在于适应值函数可直接选择为优化问题的目标函数, 算法搜索速度快, 结构简单, 计算复杂度低, 对目标函数较为光滑的优化问题有较好的效果。目前, Rudolph 从理论角度对这种算法的收敛性、收敛速度及自适应步长的引入都进行了较多研究, 但 ES 与多目标优化问题的理论分析有待开展。从应用角度, 该算法已成功用于多目标最优化设计问题, 目前这种算法的研究主要集中在自适应步长的设计及探讨多目标进化策略算法。

EP 是基于实数编码的并行搜索算法, 其与 GA 的区别在于 EP 的选择为 K-联赛选择, 突变为高斯变异策略。目前关于 EP 的研究主要是探讨 EP 与其他方法结合的混合规划方法, 应用对象是约束函数优化问题, 如 Maa 和 Shanblatt 提出两步最优化神经网络处理约束优化问题, 此方法是首先利用 EP 获得最优解的近似解作为神经网络的初始解, 然后利用神经网络获得最优解的精确值。Kim 和 Myung 提出两步混合规划算法处理约束优化问题, 即将 EP 作用于处罚函数获得较好的进化群体, 然后将处罚函数改变为增广的 Lagrange 乘子函数, 并再次利用 EP, 最终获得最优解。

模拟退火算法 SA 的思想是 Metropolis 等在 1953 提出的, 之后 Kirkpatrick 等

人于 1983 年在科学杂志上将其用于组合优化。此算法属于局部搜索算法,其基本思想是设计初始状态、初始温度及物体冷却的退温函数,以单个个体作为当前状态,在此状态的邻域中产生新状态,并按 Metropolis 准则接受稍次的新状态作为下一状态,最终达到寻优的目的。目前关于 SA 的理论及应用研究较多,理论研究已逐渐趋于成熟,应用范围已扩展到了许多领域,同时 SA 与其他智能方法的融合是 SA 能吸引许多领域的专家进行广泛研究的关键所在。由于 SA 的自身结构特性的限制,如何提高其搜索性能一直是计算智能研究者关注的焦点。近来对 SA 的研究主要集中在 SA 和其他智能算法的结合,以及如何扩展 SA 的结构并将其有效地应用于多目标优化问题。

神经网络解决优化问题的研究也在不断开展,这方面的起源研究是 Pyne 教授首次引自电路回路并用于解决非线性规划 (NP) 问题,之后又出现许多基于电路回路的模型处理 NP 问题,其中具有代表性的是 Chua 和 Lin 提出的经典非线性规划回路,以及 Hopfield 提出的 Hopfield 网络。非线性规划回路建立了一般性电路回路模型,为了增强此模型的动力学行为, Hopfield 在电路回路中引入了电容器,从而获得了 Hopfield 网络。针对已有网络存在不能保证收敛到可行解的缺陷, Walter 基于非线性规划回路的思想建立了无等式约束的约束优化神经网络模型。与此同时,两步神经网络的出现标志着神经网络处理约束优化问题取得了初步进展。至今,对于神经网络处理 NP 问题的研究主要集中在 Hopfield 网络,此网络已获得广泛应用,其理论研究也在广泛开展,但已有结果表明该网络的同步离散型模型是非稳定的。由于此网络结构的限制,致使其在 NP 问题中的应用受到极大的阻碍,困难在于构造能量函数。因此必然导致从其他角度探讨神经网络处理一般性 NP 问题。近来宋荣方等经设计能量函数,获得了一种新的神经网络处理线性约束凸优化问题,论证了此模型的稳定性。与此同时孟志青等基于静态处罚法,引入二次非线性罚函数并作为能量函数获得一种处理凸优化问题的神经网络,理论上论证了该网络在一定条件下平衡点序列收敛到最优解,有应用前景,但要求在不断增大罚因子情形下,网络产生平衡点序列,而且不能保证网络的平衡点为可行解,也阻碍了其应用。至今,神经网络模型处理优化问题的研究未能取得重大突破。

针对智能算法研究概况,本书对近年来出现的四大主要研究分支所关注的主要内容进行一系列研究,即人工免疫系统理论及应用、分形算法理论及应用、蚁群优化算法理论及应用和支持向量机,其目的在于使计算智能研究者、学者和工程人员更多地了解智能算法的研究现状,有助于智能算法不断向前发展。

第一节 人工免疫系统概述

经典智能算法不断向前发展,促使新的智能方法的不断出现,探讨新智能方法处理工程问题已成为热门话题。人工免疫系统的研究起源于 20 世纪 80 年代末期,

Farmer 作为这方面研究的先驱首次将免疫机理和人工智能结合之后,免疫系统的许多特征引起了研究者们的广泛兴趣。90年代初,免疫学的多样性机理被成功地用于遗传算法之后,陆续出现了许多与免疫有关的新的智能算法,如免疫遗传算法、克隆选择算法、模式跟踪算法、阴性选择算法、免疫网络算法、免疫网络与神经网络结合的控制算法等。这些方法已在工程领域获得了广泛应用,如模式识别、多模态函数优化、字符识别、TSP问题、加工调度、模式跟踪、自适应控制、入侵检测、故障诊断、数据分析以及机器人行为控制等。可是免疫系统的许多机理所隐含的丰富思想与工程问题结合的研究还处于初步阶段。目前在国内外,人工免疫系统理论及应用正被广泛研究,已成为人工智能领域又一研究亮点。人工免疫系统来自于对免疫系统概念、机理、特征、原理的模拟,对工程问题的解决越来越显示出了它的优越性。免疫学作为一门独立地反映人体及其他动物免疫系统运动规律的学科,对人类及其他动物的疾病防治方面作出了巨大贡献。免疫系统中免疫细胞的功能及免疫网络的动力学行为特性一直是免疫学研究的前沿问题,至今已获得突破性进展。这种系统是一种高度并行的分布式、自适应信息处理学习系统,体现了免疫系统与抗原相互作用及免疫细胞学习抗原模式结构的内在变化规律。从工程应用角度来看,免疫系统的行为特性及运行机制与工程领域中许多研究方向有着紧密的联系,如最优化、计算机安全、智能控制、故障诊断、数据分析、图像处理等等。免疫系统所隐含的丰富资源给予工程人员更多灵感,开发、研究和建立适用于工程问题的人工免疫系统,已成为工程界关注的焦点,并已在国内外形成活跃的研究氛围。目前,许多研究人员从免疫系统的不同侧面,找到了与工程问题的结合点,并已提出了许多智能方法有效地应用于现实问题。免疫系统作为复杂的信息处理学习系统,尽管其一些思想已在工程问题的解决中得到体现,但仍有大量的机理、思想方法、行为特征等有待工程界研究人员利用,并提出有效的智能工具。这不仅有助于工程问题的解决,而且也有益于激发免疫学家深入探讨免疫系统的内在运行规律,因而基于免疫系统研究和开发人工免疫系统是十分有趣及有意义的新的研究方向。

目前在人工免疫系统与工程结合过程中,主要研究方向有免疫优化算法、免疫网络、入侵检测、控制系统设计、故障诊断、免疫 Agent、机器人控制、基于免疫的计量化学等等,这些研究分支的不断深入探讨和应用丰富了人工免疫系统的内涵。深入挖掘免疫系统的内在机制,开发更多、更有效的智能系统解决工程问题是目前人工免疫系统研究的首要任务之一。在国内外,人工免疫系统理论及应用已倍受关注,其研究专家、学者主要分布在美国、中国、日本、英国、巴西、墨西哥、韩国、俄罗斯等国家,许多研究者已获得很多科研成果且被成功地应用于实际问题。但由于人工免疫系统的研究在1997年之后才开始兴起,对其开发、理论研究、应用都有待进一步探讨。

关于人工免疫系统的定义尚未达成广泛共识,在此选择莫宏伟关于人工免疫

系统的定义,即所谓人工免疫系统(从工程和科学角度讲),就是研究借鉴、利用生物免疫系统(主要是人类的免疫系统)的各种原理和机制而发展的各类信息处理技术、计算技术及其在工程和科学中应用而产生的各种智能系统的统称。从这个意义上说,人工免疫系统不仅涉及计算系统而且涉及信息处理的其他技术。开发免疫系统的任务不是激发人们对免疫系统的内在功能的运行规律进行深入研究,而是激发人们提取免疫系统中对解决工程问题有益的特征和免疫机制,并借助免疫学原理对各种运行机制进行有机组合,建立能有效解决实际问题的智能系统。

目前,人工免疫系统理论及应用已作为新的研究方向在许多国际会议上被作为专题进行了讨论,同时于2002年9月及2003年9月分别在英国肯特大学、爱丁堡大学召开了第一次、第二次人工免疫系统理论及应用国际会议,这标志着此领域已成为正式研究方向融入到人工智能领域。

第二节 分形编码概述

自20世纪80年代以来,计算机在各行各业和社会生活的许多方面得到了广泛的应用,并已被广泛应用于数据处理与数据通讯。特别是20世纪90年代以来,计算机系统的时代特征是多媒体技术应用的不断发展。简单地说,多媒体技术就是指用计算机综合处理文字、声音、图形、图像等多种媒体上承载的信息。图像是其中最主要的信息载体,因为信息的图像载体存在很多适合于人类视觉系统的优点。这一信息系统涉及图像压缩等方方面面的问题,尽管每个问题都很重要,但以较少的空间储存海量文件(如图像)的压缩是解决数据有效传输与储存的关键问题。随着多媒体应用的日益增多,如何高效、实时地压缩图像是多媒体技术中最关键的问题之一,图像压缩技术已经成为一个十分重要的研究领域。

如果从Oliver等提出PCM编码理论算起,图像压缩技术已经走过50余年的光辉历程,其间诸如预测编码(又称DPCM)、变换编码和向量量化编码等许多经典压缩编码方法被提出,并得到较为广泛的实际应用。众所周知,经典压缩编码主要依据图像本身固有的统计特性,较少利用人眼视觉系统的特性。其压缩效率一般要受信息熵的约束,不能实现很高的压缩比。然而随着图像技术的广泛应用,特别是通信的实时性对图像压缩比要求越来越高。超高倍的数据压缩对图像通信技术是一个极大的推动,但按经典压缩编码方法是难以实现的。随着感知生理—心理学的发展,人们越来越清楚地认识到视觉感知是一种宏观认识过程,人的视觉感知特点与统计意义上的信息分布有时并不完全一致,统计上需要许多信息量才能表征的某些特征对视觉感知也许并不重要。因此,从感知角度来说,详细表征这部分特征是不必要的。受此启发,人们从微观转向宏观去研究开发新的编码方法,并注重对感知特性的利用,小波编码、模型编码与分形编码等新一代编码技术自然就应运而生。

分形图像编码 (fractal image coding) 是 Bernsley 和 Sloan 于 1988 年提出的美国专利技术, 源于他们对迭代函数系统的研究, 对几幅图像的分形编码获得了难以置信的超高压缩比 (10000:1)。尽管事实证明这种方法是不实用的 (编码时间太长, 且需要人机交互, 对操作者有较高要求), 但它毕竟为图像压缩提供了一条与以往完全不同的新思路。实际上, 目前的实用分形图像编码正是在此基础上发展起来的。在分形编码实用化的研究中, Jacquin 迈出了实质性的一步, 在其发表于 IEEE 刊物的著名论文里提出了分形块编码 (Fractal Block Coding), 这是一种基于方块划分的计算机自动分形编码算法, 其理论基础被称为 PIFS (partitioned IFS)。分形块编码算法建立在图像局部自相似的基础上, 在最简情形下, 图像被分割成大两小两类子块, 子图像互不重叠且覆盖整幅图像。然后对每个小方块, 在图像中搜索与之最仿射相似的大方块, 这样每个小方块由与其匹配大方块的仿射变换 (大小比例重定、旋转、反射与亮度变换) 来近似。这些作用于大块的变换以分块方式定义一个作用于整幅图像上的分形变换, 其不动点逼近待编码图像。这就给出了图像数据的一种自相似描述, 解码后的图像呈现出一定的自相似结构。

纵观大量分形图像编码文献, 在上述基本原理的框架内, 在提高编码质量和加快分形编码等方面, 许多新观点和大量改进算法被提出, 对它们进行准确的分类是困难的。但总的来说, 大多数现有分形编码文献都是围绕下面的基本问题展开的, 或者说大多数分形编码方案的区别主要反映在下列几个方面:

(1) 图像分割。尽管固定方块分割方案实现简单, 也有分割方案占有零信息量的优点 (即无需存储或传输分割信息), 但编码质量不理想, 因为方块尺寸必须事先指定且不依赖于待编码图像, 也没有利用图像内容, 不能自适应于不同的图像, 也不能自适应于同一图像的不同细节 (如飞行中的飞机照片, 飞机与蓝天的细节显然不同)。二叉树分割方案虽然有一定的图像适应性, 且描述二叉树分割的成本也很小, 但它对图像的适应性仍然是有限的。因此, 研究比二叉树分割的图像适应性更强的自适应分割方案是一个重要的问题。

(2) 虚拟码本构成。从分形图像编码的迭代函数系统观点来看, 这就是如何构造 D 块 (domain block) 池的问题。与向量量化 (VQ) 编码一样, 合适的虚拟码本对编码时间和编码质量都有很大影响。以固定方块分割为例, 增大虚拟码本的容量 (如以小步长生成 D 块池、同时考虑八个等距变换), 编码质量自然会提高, 但这必须以牺牲编码时间和压缩比为代价——虚拟码本越大, 匹配搜索的范围越大, 编码时间自然越多, 同时也需要更多的比特数来指明最佳匹配块的位置。在虚拟码本构成方面, 许多 VQ 码本构成策略被移植到分形图像编码中。

(3) 亮度变换类型。亮度变换是指作用于码本块的变换, 基本分形图像编码的亮度变换仅仅实现亮度调整与亮度补偿, 即每个 R 块用码本块和常值亮度块的线性组合来编码 ($R \approx sD + o1$)。显然, 按这种方式构造的变换具有结构简单、计算方便的优点, 但是其编码能力是有限的。没有理论结果显示这种选择就是最优的。

(4) 变换参数的量化、比特分配与熵编码。在分形编码中,一幅图像是用一个使图像近似不变的压缩变换来表达的,而变换又是由表达它的参数确定的。为了进一步实现图像数据的压缩,表达压缩变换的亮度调整参数和亮度偏移参数必须进行量化处理,并对量化参数进行合理的比特分配。此外,熵编码可以进一步改进分形编码效率。

(5) 分形解码问题。尽管解码快是分形图像编码公认的优点之一,但在有些实时性要求高的应用领域,更快的解码是必需的。因此,更快速分形解码也得到了人们的关注。另一方面,在另外一些应用领域(如窄带宽传输、制作动画),可能需要能够以某种方式控制解码过程,如渐进解码。对于后一方面,目前只有本书作者对此进行了探索,并在发表于 IEEE 刊物的论文中首次引入“可控分形图像解码”的概念,提出了一个新颖的分形解码算法,在不修改现有分形编码过程的前提下实现了分形渐进解码。

(6) 基础理论研究。迭代函数系统逆问题的研究是其核心问题,尽管人们已尝试各种方法以解决这个问题,但目前仍然没有得到理想解决。我们知道,压缩图像的分形文件就是一个压缩变换的描述,对于如何寻求这样的压缩变换,拼贴定理是目前依据的理论基础。但是,拼贴定理只是给重构误差提供一个上界(体现为所谓的拼贴误差),这个上界极小化并不意味着重构误差的极小化。因此,目前的分形编码方案远不是最优的。当然,相关的基础理论问题并不止这个逆问题。例如,更好的数学框架、解码收敛性、解码误差控制以及分形编码机理等问题。

(7) 加快编码过程。这主要涉及匹配搜索问题,因为在分形编码阶段,匹配搜索占据了主要的编码时间。全搜索固然能够得到最好的编码质量,但全搜索的运算量大是不争的事实。加快分形编码速度的关键是设计好的搜索方案(与虚拟码本的构成密切相关),这是分形图像编码实用化的核心问题。

(8) 混合分形编码。大多数纯粹基于分形理论的编码方法与目前广泛使用的编码方法相比,还缺乏竞争力。但是,分形编码与其他编码方法(如向量量化、小波、神经网络等)结合的混合编码方法已经取得巨大成功。

第三节 蚁群优化概述

在自然界中,像蚂蚁这类社会性动物,虽然个体的行为极其简单,但由这些简单个体所组成的蚁群却表现出极其复杂的行为特征:蚁群有能力在没有任何提示下找出从蚁巢到食物源之间的最短路径。此外,蚁群还能适应环境的变化,如在蚁群经过的路线上出现障碍物时,蚂蚁能够很快找到新的最优路径。

蚁群是如何完成这些复杂任务的呢?仿生学家经过大量地观察、研究发现,蚂蚁在寻找食物源时,能在其经过的路径上释放一种蚂蚁特有的分泌物——信息激

素 (pheromone)^①, 使得一定范围内的其他蚂蚁能够感觉到这种物质, 且倾向于朝着该物质强度高的方向移动。因此, 蚁群的集体行为表现为一种信息正反馈现象: 某条路径上走过的蚂蚁越多, 其上留下的信息素也越多 (当然, 随时间的推移会逐渐蒸发掉一部分), 后来蚂蚁选择该路径的概率也越高, 从而更增加了该路径上信息素的强度。蚁群这种选择路径的过程被称之为自催化行为 (autocatalytic behavior), 由于其原理是一种正反馈机制, 因此也可将蚁群的行为理解成所谓的增强型学习系统 (reinforcement learning system)。

显然, 自然界中蚁群的行为具有显著的特征: (1) 蚂蚁能察觉小范围内的状况并判断出是否有食物或其他同类的信息素的迹; (2) 能释放自己的信息素; (3) 信息素的强度会随时间而逐步减弱。由于自然界中的蚂蚁基本没有视觉, 既不知去何处寻找和获取食物, 也不知发现食物后如何返回自己的巢穴, 因此它们仅仅依赖于同类散发在周围环境中的特殊物质——信息素的迹来决定自己何去何从。有趣的是, 尽管没有任何先验知识, 但蚁群还是有能力找到从蚁巢到食物源的最优路径, 甚至在该路线上出现障碍物之后, 它们仍然能很快找到新的最佳路径。

正是从自然界中蚁群的觅食行为中受到启发, 意大利学者 M Dorigo, V Maniezzo 和 A Colomi 利用该过程与著名的旅行商问题 (traveling salesman problem, TSP) 之间的相似性进行比较和研究, 于 20 世纪 90 年代初提出了一种新的模拟进化算法——蚂蚁系统 (ant system, AS)^②。实验结果表明该算法具有较强的鲁棒性和发现较好解的能力, 但同时也存在一些缺陷, 如收敛速度慢、易出现停滞现象等。针对该算法的不足, 一些学者提出了许多改进的蚁群优化算法, 如蚁群系统 (ant colony system, ACS)、最大-最小蚂蚁系统 (max-min ant system, MMAS)、最优保留蚂蚁系统 (ant system with elitist, AS_{elitist}) 和基于排序的蚂蚁系统 (rank-based version of ant system, AS_{rank}) 等。这些改进的算法很大程度上消除了搜索中的停滞现象, 极大地提高了算法的性能。近年来, 一些学者提出了蚁群优化元启发式 (ant colony optimization meta heuristic, ACO-MH) 这一求解复杂问题的统一框架, 该框架为蚁群优化算法的理论研究和设计提供了技术上的保障。

虽然蚁群优化算法出现才短短十几年时间, 但已在许多领域得到了广泛的应用。M Dorigo 等人首先将 AS 算法应用于 30 个城市的 TSP, 在选择适当的参数后, AS 算法总能收敛到问题的最优解。V Maniezzo 等人将 AS 算法应用于二次分配问题 (quadratic assignment problem, QAP)。另外, Ambardella, Taillard 和 Stutzle 也发表了一些用蚁群优化算法求解 QAP 的文章, 目前, 蚁群优化算法已经是求解 QAP 最有效的算法之一。Costa 和 Herz 提出了增强的 AS 算法, 该算法在求解图着色问题 (graph coloring problem, GCP) 时的结果完全可以和其他启发式算法相媲

① 鉴于该词目前尚未找到更确切的译法, 本文以后将称之为信息素。

② 关于该算法名称的译法有多种, 如蚂蚁算法、蚁群算法和蚁群优化算法等, 本书以后统称为蚁群优化算法。

美. Bullnheimer, Hartl 和 Strauss 用 AS_{rank} 算法求解车辆路径问题, 并取得了和最优解非常相近的结果. Gambardella, Taillard 和 Agazzi 对 VRP 的研究, 并对一些标准基 (benchmark) 问题的最优解的求解速度有所提高. A Colomi 等人首先将 AS 算法应用于作业调度问题 (job shop scheduling problem, JSSP), 结果表明该算法同 state-of-the-art 算法相比没有任何优势, 但在求解流式 JSSP (flow job-shop scheduling problem) 时, 实验结果表明 MMAS 算法优于模拟退火算法和禁忌搜索算法.

近几年, 蚁群优化算法还在函数优化、系统辨识、机器人路径规划、数据挖掘、大规模集成电路 (VLSI) 中的综合布线设计、网络路由等领域取得了引人注目的成绩. 特别是该算法在网络路由中的应用受到越来越多学者的关注, 并提出了一些新的基于蚁群优化算法的网络路由算法. 同传统的路由算法相比, 该算法在网络路由中具有信息分布性、动态性、随机性和异步性等特点, 而这些特点正好能满足网络路由的需要.

对蚁群优化算法的研究虽然刚刚起步, 但初步的研究结果已显示出该算法在求解复杂优化问题 (特别是离散优化问题) 方面的优越性. 尽管一些思想尚处于萌芽时期, 但人们已隐隐约约认识到, 人类诞生于大自然, 解决问题的灵感似乎也应当来自于大自然. 因此, 蚁群优化算法正在受到越来越多人的注意和研究, 应用范围也开始遍及许多领域. 从当前可以检索到的文献情况来看, 研究和应用蚁群优化算法的学者主要集中在比利时、意大利、英国、法国和德国等欧洲国家. 日本和美国在这两年内也开始启动对蚁群优化算法的研究. 目前, 蚁群优化思想在启发式方法范畴内已逐渐成为一个独立的分支, 在有关国际会议上多次作为专题加以讨论. 1998 年在比利时布鲁塞尔大学召开了第一届蚁群优化国际研讨会, 2000 年、2002 年仍在原地召开了第二、三届会议 (Ants'2000, Ant'2002).

尽管蚁群优化的严格理论基础尚未奠定, 国内外的有关研究仍停留在实验探索阶段, 但从当前的应用效果来看, 这种模仿自然生物的新型系统寻优思想无疑具有十分光明的前景, 更多深入细致的工作还有待于进一步展开.

第四节 支持向量机概述

学习是一切智能系统最根本的特征, 机器学习是人工智能最具智能特征、最前沿的研究领域之一. 基于样本的机器学习问题是现代智能算法技术的一个重要分支, 它模拟了人类从实例中学习归纳的能力, 主要研究如何从一些观测数据 (样本) 中挖掘出目前尚不能通过原理分析得到的规律, 并利用这些规律去分析客观对象, 对未知数据或无法观测的新现象进行预测和判断. 统计学在解决这类机器学习中起着基础性的作用, 但是, 传统的统计学所研究的是渐近理论, 即当样本趋向于无穷大时的极限特性. 然而, 现实应用当中, 样本数目通常是有限的, 有时样本的获

取甚至是非常困难的,例如特殊故障模式下数据的获取就很困难.因此研究在小样本数据量下的统计学习规律是一个非常有实用价值的问题.

以 V N Vapnik 为代表的学习过程理论分析学派早在 20 世纪 60 年代就开始研究有限样本情况下的机器学习问题.由于涉及艰涩的数学理论和方法论上的重大革新,90 年代以前并没有提出能够将其理论付诸实践的方法,加之当时正处于神经网络的其他学习方法飞速发展时期,因此这些研究并没有得到充分的重视.90 年代初期,有限样本的机器学习理论逐渐成熟起来,形成了一个较为完善的理论体系——统计学习理论 (statistical learning theory, 简称 SLT).同时神经网络 (ANN) 等较新兴的启发式机器学习方法的研究则遇到了推广能力差的困难,如神经网络结构的确定、过学习和欠学习、局部收敛等问题.目前 SLT 与在此基础上发展起来的 SVM 机器学习方法已经成为机器学习领域最新的研究热点.

作为统计学习理论的重要创始人和支持向量机的发明者, V N Vapnik 先后于 1995 年和 1999 年两次出版了该研究领域的重要著作《The Nature of Statistical Theory》,该书重点研究了统计学习理论中的四个方面的重要问题,即经验风险最小化下学习过程一致性的充分必要条件、收敛速度的非渐近理论、学习机推广能力的控制策略以及如何构造理想的学习机.此文献对推动统计学习理论和支持向量机的研究起到了十分重要的作用.我国学者张学工将该文献译成了中文,为推动 SLT 和 SVM 在国内的研究和应用起到了积极的作用.

SVM 是一种崭新的机器学习方法,在理论上具有很突出的优势.但与其理论研究相比,应用研究则相对比较滞后,目前只有较有限的实验研究报道. SVM 的应用应该是一个大有作为的方向.目前,在模式识别方面最突出的应用研究是贝尔实验室对美国邮政手写数字库进行的实验,用 SVM 方法得到的识别结果明显优于决策树和多层神经网络.相关的应用还包括 SVM 与神经网络相结合对笔迹进行在线适应, MIT 用 SVM 进行的人脸检测实验也取得了较好的效果,可以较好地学会在图像中找出可能的人脸位置.其他有报道的实验领域还包括文本识别、人脸识别、三维物体识别、遥感图像分析等.可见, SVM 的应用大都集中于图像处理领域,已有文献给出了将 SVM 机器学习方法引入工业领域进行故障诊断的试验,利用基于 SVM 的二叉树多类分类方法有效地解决了柴油机振动信号的故障识别问题.

由于 SVM 算法是从两类分类问题推导出来的,在解决像故障诊断等典型的多类分类问题时会遇到困难.将 SVM 机器学习方法延伸到多类分类问题目前还处于初步研究阶段,已经提出的解决这个问题的思路主要有两大类: (1) 构造多个 2 类分类器组合起来完成多类分类,该思路源于传统的模式识别方法解决多类问题.根据训练样本组成的不同,又分成“one against one”和“one against all”两种类型,前者的每一个 SVM 的训练样本是由两个不同类别的样本组成,需要构造 $(k(k-1)/2)$ 个 SVM 才能完成整个分类任务, k 为类别数;后者的每个 SVM 的训练样本都由

全部样本组成,需构造 k 个支持向量机才能完成. 这类思路简单有效,但存在无法识别的阴影区域,而且重复训练的样本较多. (2) 只使用一个 SVM 机实现多个分类输出 (all together), 这种思路涉及十分复杂的优化问题, 训练样本数目相对较大时, 需要很长的运算时间, 而且分类精度不很理想.

SVM 是一个富有广阔发展前景的研究领域, 针对其在多类别分类方面的研究, 本书探讨了 SVM 在故障诊断中的应用, 并以柴油机振动信号的故障诊断为例, 利用基于 SVM 的二叉树多类分类方法, 成功地实现了柴油机故障检测和故障原因判断, 并通过大量的实验, 为分析 SVM 理论中有关参数的特性和选择依据提供了一种新方法. 同时还研究 SVM 回归算法的基本实现方法, 给出运用 SVM 机器学习方法进行故障趋势预测的方法, 通过对 “Tennessee Eastman” 工厂的实际数据进行仿真, 体现 SVM 方法进行故障趋势预测与控制的优越性.

第一部分 免疫优化 及免疫网络算法理论和应用

第二章 免疫学基本理论 及人工免疫系统概论

第一节 免疫学基本概念及原理

一、免疫学基本概念

免疫学作为一门独立地反映人体及其他动物免疫系统运动变化规律的学科，对人类作出了巨大贡献。免疫系统是一种高度并行的分布式、自适应信息处理学习系统，其结构及行为特性极为复杂，对其内在运行规律的认识，免疫学家们仍在作不懈地努力。这种系统的作用在于识别自我及非自我物质，清除和防御外来入侵的病毒物质或分子。其主要在巨噬细胞、抗原呈递细胞、主要组织相容性复合物及 B、T 细胞的作用下，通过抗体识别抗原的模式结构及抗体自身进化的方式完成匹配抗原的任务。抗体与抗原的作用机制反映了免疫系统属于一种进化的防御系统，这种进化方式启发人们开发新的计算智能工具解决不断涌现的复杂非线性问题。为了便于从免疫系统的机理出发，开发智能工具处理工程问题，在此对在人工免疫系统中所用到的主要免疫学基本概念及原理加以概述，并进行理论分析，详细内容可参阅文献 [1]。

免疫 免疫系统受到外来病毒或分子刺激后所产生的一系列复杂变化。免疫的作用是识别和排除非己物质，以维持机体的生理平衡。正常情况下，免疫应答的结果对机体有利，起到免疫保护、免疫稳定和免疫监视等生理性保护作用。在免疫缺陷及功能失调或免疫应答过程不足时，则可造成病理损伤或过敏反应。

抗原 通常指外来感染性物质或分子，以及胚胎期末在与免疫系统接触中发生改变的自身物质，即自身抗原。抗原由载体和半抗原组成，半抗原又称为抗原决定基或表位，一个抗原可有一个或多个不同的表位，其表位成份数目和空间构型决定抗原的特异性。抗原的表位与机体免疫细胞表面的受体相结合引起免疫应答，产生抗体和（或）致淋巴细胞等免疫物质，同时在体内存在相应抗体和（或）致淋巴细胞与抗原发生特异性结合，使抗原得以清除。抗原具有两种性能：（1）刺激机体产生免疫反应；（2）与相应抗体和（或）致淋巴细胞发生特异性结合。

B 细胞 产生于骨髓并在骨髓中发育成熟的淋巴细胞。B 细胞经由激活，分化为多个浆细胞，每一个浆细胞能分泌多个相同类型的免疫球蛋白，B 细胞的作用是产生抗体，并对入侵者作出应答。B 细胞表面载有相同类型的抗体，当 B 细

胞被激活后,其表面的抗体识别和纠缠其他特别的外侵者,抗体的产生和纠缠暗示将摧毁、阻塞及驱除那些被缠的物质。

抗体 B细胞接受抗原刺激后,所分泌的具有免疫功能,并能与抗原发生特异性结合的免疫球蛋白。抗体的基本结构是一种“Y”型的四肽链,它由两个完全相同的轻链和两个完全相同的重链及二硫链构成。在抗体的结构上,有氨基酸数量和排列较保守的恒定区,另有因抗体不同而不同的可变区。在可变区中,有一部分为较容易改变氨基酸排列次序的高变区,这种高变区体现了抗体识别抗原的特异性及多样性。

自我抗体 免疫系统中随机产生的B细胞所携带的抗体。

T细胞 骨髓中分泌的部分淋巴液被分配到胸腺,这些淋巴液在胸腺中发育成熟后则为T细胞。T细胞的作用在于调整其他细胞的行为和袭击体内的感染性细胞。这种细胞主要包含辅助性T细胞 T_H 、杀伤性T细胞 T_K 和抑制性T细胞 T_S 。 T_H 细胞帮助激活B细胞和其他T细胞。 T_K 细胞能驱除微生物入侵者、病毒及癌细胞,一旦其被激活和被约束于其他组织,它们就注射有害物质进入其他细胞,繁殖它们的表面组织和导致这些组织被毁灭。 T_S 细胞对维持免疫应答起重要作用,抑制其他免疫细胞的活动(即控制免疫系统的行为,抑制其他细胞),没有 T_S 细胞将导致免疫细胞松弛及过敏性反应和自治免疫疾病。

免疫细胞 由来自骨髓的淋巴细胞和包含颗粒状化学物质的白细胞组成。在骨髓里,许多白细胞成熟,系统的细胞移入巡逻的组织,这些细胞中,有些负责通常的防御,其他的被训练以便防止非常具体的传染疾病。

淋巴细胞 由B细胞、T细胞和NK自然杀伤细胞等组成。

巨噬细胞 一种大的且能阻塞和消化微组织和抗原的白细胞,其作用在于把抗原分泌为颗粒状,并将这些物质送给淋巴细胞。

抗原呈递细胞 能摄取、加工处理抗原,并将抗原肽呈递给淋巴细胞的一类免疫细胞,其主要作用在于将抗原肽提供给T细胞识别。

骨髓 一种软组织造血器官,它包含在最长的骨里,负责免疫细胞的繁殖。

免疫记忆 免疫系统的一个重要特征是好交往性,它不仅能记忆已出现抗原的结构,而且对相同抗原的再次入侵或相似抗原的出现作出快速反应,能成功地毁灭被识别的抗原。免疫记忆是免疫系统的重要特征,有助于加快再次免疫应答过程。

亲和力 抗体的表位与抗原的对位的匹配程度。

二、免疫系统组成及结构分析

免疫系统是一种由众多细胞、分子和组织等子系统构成的复杂系统,这些子系统之间存在着复杂的相互联系,具有能识别“自己”和“非己”,清除和消灭异物的

功能。免疫系统的构成要素如图 2-1 所示,此系统由先天性免疫系统和自适应免疫系统组成。先天性免疫系统是一种与生俱有的天然防御系统,不具有特异性,具有识别一定微生物和立即摧毁这种微生物的能力。它包含一种被称为补体的血蛋白质,这能帮助和增加抗体的活动能力,尤其是首次遇上的许多传染物质(抗原)。这种系统的主要作用在于能辨析自我和非自我,参与自我和非自我辨析,对推动自适应免疫起着重要的作用。自适应免疫系统是后天形成的一种防御系统,能自适应学习外来入侵病毒物质或分子的模式结构,清除或中立这种分子或物质。

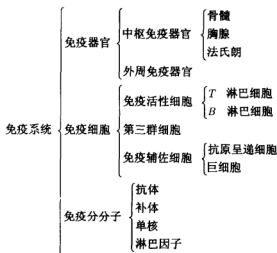


图 2-1 免疫系统的组成

免疫系统具有较强的免疫防御功能,即免疫防护、免疫稳定及免疫监视。其中免疫防护具有抗病原侵袭的作用;免疫稳定具有消除损伤或衰老细胞及免疫调控的作用;免疫监视具有消除癌变细胞或病毒感染细胞的功能。免疫的目标是摧毁来自外界的细菌或其他入侵者。

在抗原激励下,免疫系统的运行机制可简要地归纳为以下四步,图形描述如图 2-2 所示。

(1) 巨噬细胞分化抗原为颗粒状物质,主要组织相容性物质与这种颗粒状物质结合形成粘液物质,抗原呈递细胞将这些物质呈递到巨噬细胞的表面。

(2) 通过识别的途径,被激活的 T 细胞分化和分泌淋巴因子或化合物信号,并驱动免疫系统的 B 细胞应答。

(3) B 细胞对来自激活的 T 细胞的信号作出反应,但不像 T 细胞那样, B 细胞能自由地识别部分抗原。即使无 T 细胞的作用, B 细胞仍对抗原作出反应, T 细胞仅对 B 细胞应答抗原起到调控作用。

(4) 当 B 细胞被激活时, B 细胞就分化和繁殖,这种细胞能分泌出抗体蛋白

质, 这些抗体能游离出来, 抗体通过缠住已发现的抗原, 并中立这些抗原以至毁灭它们, 其他多余的 T 细胞和 B 细胞变为记忆细胞。

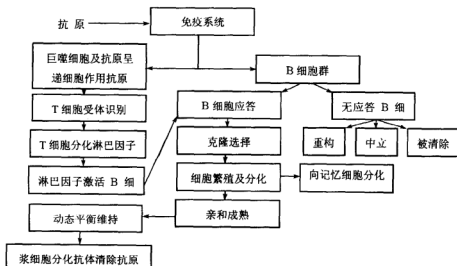


图 2-2 免疫应答的机制

三、免疫应答过程分析

(一) 免疫应答

免疫应答是指抗原进入机体后, 刺激免疫系统所发生的一系列复杂的变化。这种应答包含先天性免疫应答和自适应免疫应答。在此应答过程中, 抗原呈递细胞将抗原颗粒传递给 T 细胞, T 细胞通过识别抗原后自身进行活化、增殖, 经由 T 细胞的作用, B 细胞分化、繁殖、成熟及自我调节促成 B 细胞自适应学习抗原模式。先天性免疫应答是机体在遗传基因作用下, 在长期的发育和进化中形成的一种应答方式, 它的特点是具有相对稳定性, 具有防御微生物入侵机体的作用, 同时是任何有生命的事物生来就有的天然保护系统。由于此类应答并不需后天培育, 对病毒的类型无特殊的针对性, 因此称为非特异性免疫应答。自适应免疫应答, 又称为特异性免疫应答, 是机体接受抗原刺激后, 机体自组织学习抗原的应答过程, 其包含细胞免疫和体液免疫。

细胞免疫是指 T 细胞介导的免疫应答, 这种应答过程主要通过巨噬细胞、抗原呈递细胞、主要组织相容性复合物及 T 细胞共同参与及相互协调完成。

体液免疫是免疫应答的核心内容, 它指抗体介导的免疫应答。这种应答由初次应答和再次应答组成, 参与应答的细胞主要是 B 细胞、T 细胞、巨噬细胞和抗原呈递细胞, 这种应答的机理可由图 2-3 获知。对于抗原初次入侵机体时, 免疫系统

中少量的 B 细胞被激活, 这些 B 细胞分化出大量的 B 细胞, 其中一些 B 细胞作为记忆细胞被存于免疫系统中, 另一部分则繁殖大量的克隆细胞并经由亲和突变产生多样的免疫细胞。这些细胞中, 识别抗原能力强的部分细胞中立抗原, 识别能力弱或产生自反应的细胞被清除, 其余的细胞作为记忆细胞存于免疫系统, 这种应答称为初次应答。当相同或相似的抗原入侵机体时, 免疫系统的记忆细胞立即对抗原产生应答, 此时免疫细胞数目增加较快, 抗原被清除也较快, 这种应答称为再次免疫应答。

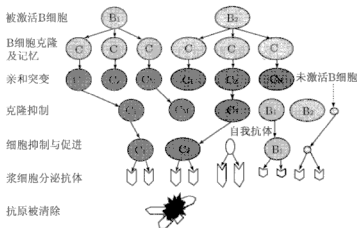


图 2-3 体液免疫应答原理

从体液免疫应答过程分析可看出, 由于识别抗原能力强的免疫细胞被激活参与进化, 因而免疫细胞的选择具有确定性。另一方面, 由于被抗原激活的 B 细胞表面的抗体与抗原不需要完全匹配, 仅需匹配程度在一定的范围, 这说明免疫细胞的存活与否具有随机性, 因此选择存活的免疫细胞具有随机性。这两方面因素表明免疫系统中免疫细胞的选择具有确定性与随机性的特征。

(二) 免疫应答具有如下的特征

结合体液免疫应答过程, 免疫应答的特征可概述如下, 其中 (1)~(3) 为文献 [2] 的叙述:

(1) 个体免疫的特异性: 不同的个体具有不同的免疫系统, 特定的免疫系统具有特殊的免疫能力及脆弱性。

(2) 自我检测能力: 对机体从未遭受的疾病作出检测及反应。

(3) 分布式及免疫耐受: 细胞遍及体内, 并且没有中心控制, 能容忍识别能力弱的细胞。

(4) 自我与非自我辨别能力: 经由抗体的识别, 机体中的细胞和分子被分类为自我物质及非自我物质, 并通过自我物质进化自身模式学习非自我物质的结构。

(5) B 细胞选择性应答: 应答抗原能力强的 B 细胞被选择分化和繁殖, 形成无性繁殖系的克隆群。

(6) 基因变异及多样性: 繁殖的细胞的基因经由随机重构及基因块重组产生种类繁多的细胞, 获得识别能力更强的细胞, 提高免疫系统防御能力。

(7) 细胞的抑制和促进: 高亲和力及低浓度的抗体得到促进, 高浓度及低亲和力的抗体被抑制。

(8) 免疫细胞群体规模动态调节: 免疫系统中克隆细胞总数被调整, 有些细胞的克隆数增大, 而另一些细胞的克隆数却减少, 总数在变化, 此由辅助性 T_H 、 T_S 细胞协同完成。

(9) 自学习和自组织: 通过免疫细胞的选择、克隆及突变规则, 以及 T 细胞的免疫调节和自我抗体的随机产生, 促成免疫系统自组织学习抗原模式, 并最终清除抗原。

(10) 免疫记忆: 抗原刺激免疫系统, 识别能力强的免疫细胞将记忆抗原的模式, 并通过学习方式获得识别能力强的记忆细胞。对于相同或相似抗原的出现, 记忆细胞将作出快速反应。

(三) 免疫学习和群体多样性

在依赖于 T 细胞的体液免疫中, 被激活的 B 细胞群体通过其表面的抗体的基因随机变异产生多样的 B 细胞, 大量的高亲和力 B 细胞被选入记忆池中, 低亲和力及高浓度的免疫细胞被抑制。经细胞繁殖后, B 细胞表面的抗体遭受变异, 抗体的变异受到严格限制, 即亲和力越高, 突变的可能性越小, 反之, 则突变的可能性越大, 突变的目的在于提高抗体识别抗原的能力。

免疫系统的自适应学习特征表明免疫系统能产生多样的抗体监测及摧毁抗原。抗原约有 10^{16} 种类型, 同时人体约有 10^5 种基因。基因通过随机组合构成基因块, 基因库则由基因块构成, 抗体由基因块随机组合而成, 因此尽管免疫系统中仅有约 10^5 种基因, 但基因块的随机组合却产生约 10^{15} 种不同类型的抗体, 从而免疫系统能应答各种类型的抗原, 即免疫系统具有多样性特征。这些数据摘自文献 [1,11]。

通常增强多样性的方法有单点突变、超突变、基因逆转及基因块互换等。免疫学习的获取可通过几种方式获得: 重复遭受抗原袭击、抗体分子的亲和成熟及免疫网络调节。

四、免疫学基本原理概述

解释体液免疫应答过程的代表性原理是奥地利免疫学家伯内特 (N K Burnet) 于 1957 年提出的细胞克隆选择学说, 以及著名的美国免疫学家耶纳 (N K Jenner) 于 1974 年提出的独特型免疫网络假说。现将体液免疫过程的机理整理归纳如下。

(一) 克隆选择原理

机体在胚胎期, 由于遗传和免疫细胞在增殖中发生基因突变, 形成了多样性的免疫细胞。这些细胞经分化后, 部分细胞变为记忆细胞, 另一部分细胞不断增殖形成无性繁殖系, 这种无性繁殖系被称为克隆。每一种抗原侵入机体后, 机体内都能选出识别相应抗原的免疫细胞, 使之被激活、分化和繁殖, 产生大量具有特异性的抗体, 这些抗体中立抗原并最终清除抗原, 这叫细胞克隆选择学说。克隆选择原理中包含着免疫细胞应答抗原的几种机理, 即应答抗原能力强的免疫细胞被确定性地选择进行应答, 在此称为克隆选择。被选择进行应答的免疫细胞依据其应答抗原的能力强弱, 繁殖一定数目的克隆细胞, 免疫细胞繁殖克隆的数目与其亲和力成正比, 这种机制称为细胞克隆。部分克隆细胞变为记忆细胞或长寿细胞, 被保存于免疫系统中并更新已有的低亲和力的记忆细胞, 在此称为记忆细胞获取, 记忆细胞产生免疫记忆。其余细胞成熟为浆细胞并分泌抗体分子, 在这细胞成熟过程中, 克隆细胞在其母体的亲和力影响下, 按照与母体亲和力成反比的概率对抗体的基因多次重复随机突变及基因块重组, 进而产生种类繁多的免疫细胞, 并获得大量识别抗原能力比母体强的 B 细胞。这些识别抗原能力较强的细胞能有效缠住入侵抗原, 这种现象称为亲和成熟。此机理主要由抗体的突变完成, 即称为亲和突变, 而突变受其母体的亲和力制约, 因此亲和成熟过程本质上是克隆细胞不断重复遭受变异的过程。突变方式有: 单点突变、超突变及基因块重组、基因块反序、基因块替换 (即用随机产生的基因取代基因块中所有基因) 等。克隆选择原理解释了免疫细胞应答抗原, 并最终清除抗原的相互作用关系, 其作用机制如图 2-4 所表示。

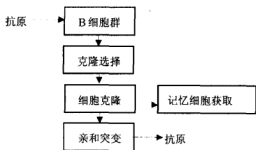


图 2-4 克隆选择原理

(二) 独特型免疫网络原理

在免疫系统中, 通过抗原的对位与抗体的表位以及抗体之间的表位与对位进行识别与被识别。抗体不仅识别抗原, 同时又识别其他抗体和被其他抗体识别, 因此抗体具有识别和被识别的特性。抗体表面的表位又称为独特型或独特型 (idiotypic) 抗原决定基, 其作用在于被其他抗体识别。通过抗体表面的受体, 抗体识别抗原, 抗体与抗体之间相互识别, 并形成了独特型免疫网络。

在此网络中, 被识别的抗体受到抑制, 识别抗原及其他抗体的抗体得到促进和增殖, 这种机制便构成了独特型免疫网络调节。免疫网络调节主要反映了免疫系统中抗体分子之间相互协调、相互促进及相互抑制关系, 这种网络即使在没有抗原的

作用下也能进行,因此又将能被其他抗体识别的抗体称为抗原的内影象。同时网络调节能使网络中抗体的总数目获得控制,并调节各种类型的抗体在免疫系统中的数目,使所有抗体的数目达到总体上平衡。当抗原入侵免疫系统时,这种平衡遭到破坏,应答抗原能力强的B细胞进行增殖,并导致免疫应答。待抗原被清除后,依赖于免疫网络调节使抗体数目达到新的平衡,这种现象称为免疫调节。

独特型免疫网络原理主要包含两种抗体作用机制,即克隆抑制及抗体的动态平衡维持(简称为动态平衡维持)。

克隆抑制是指突变的克隆群中相似及相同的抗体被确定性地抑制,促成产生具有相似度较低及细胞种类较多的免疫细胞群,这种机制类似于进化论中的小生境技术,起到维持群体多样性的作用。同时这种机制有利于减轻动态平衡维持中抗体的选择压力。

动态平衡维持使母体群(即当前免疫细胞群体)与克隆细胞群共同竞争,依据免疫细胞的浓度及亲和力按概率方式随机选择存活的免疫细胞。这种机制主要体现了抗体之间相互抑制和相互促进机理,即亲和力高而浓度低的抗体受到鼓励,反之则受到抑制。另外,免疫系统随时产生自我抗体插入存活的抗体群,以便增强抗体群多样性及调节免疫系统中抗体数目。

从免疫网络理论的描述获知,其包含的两种机制在免疫网络中起重要作用,克隆抑制能缩小抗体群的规模,促成抗体进化的作用,使得高亲和力或识别其他抗体能力强的抗体得到保存。动态平衡维持使免疫系统中抗体的总数目获得控制,同时多种抗体的数目达到平衡,相似或相同抗体出现的数目较少以及自我抗体的出现增强了免疫网络的动态平衡,这两种机制在体液免疫应答中起到调节免疫细胞群的作用。

第二节 人工免疫系统概述

一、人工免疫系统发展概况

随着科学技术及工程应用不断深入发展,智能方法正作为人工智能的重要研究分支向前快速发展。在此计算领域中,受生物免疫机制的启发,出现了继神经网络和进化计算之后的又一新研究方向,即人工免疫系统理论及应用^[2~8]。

免疫系统是一种复杂的分布式信息处理学习系统,这种系统具有免疫防护、免疫耐受、免疫记忆、免疫监视功能,尤其具有较强的自适应性、多样性、学习、识别和记忆等特点^[1]。这些功能和特点的结合给予研究人员较多的灵感,促成许多学者建立了基于免疫机理的智能方法,解决大量的非线性科学问题。免疫系统的作用在于识别和辨别自我与非自我物质,通过淋巴细胞学习非自我物质的模式并最终清除异己物质。这种学习过程本质上是一种不断进化自身模式的过程,这种运行机

制暗示了免疫系统的许多特征极大地有利于非线性工程问题的解决。

自 1986 年 Farmer 首次^[9]将免疫机理和人工智能结合起来,免疫系统的许多特征在工程领域获得了广泛的应用。20 世纪 90 年代初,免疫学的多样性机理被成功地用于遗传算法之后,一些作者获得了几种类型的免疫遗传算法,这些算法对遗传算法作了较大改进。1999 年之后,de Castro 和 Gaspar 分别从克隆选择原理出发建立了克隆选择算法及模式跟踪算法,同时 Dasgupta 等利用免疫系统中的负选择机制设计了一种阴性选择算法,并用于入侵检测问题。这些算法的出现标志着免疫学的原理及特征解决实际问题具有很大潜力。至今,开发免疫系统,提出新智能方法解决工程问题已是计算智能研究的前沿,已有的许多基于免疫的智能方法在工程领域已获得广泛应用。例如 (1) 免疫遗传算法^[9~38]及基于免疫学原理的算法^[39~50]用于模式识别、多模态函数优化、字符识别、TSP 问题、加工调度、模式跟踪、自适应控制、入侵检测、故障诊断及计量化学等;(2) 模拟免疫系统的免疫网络原理可获得免疫网络算法^[51~61]解决聚类、分类、数据挖掘等;(3) 基于免疫学原理或免疫特征与遗传算法结合的算法解决多目标优化问题^[62~66]。可是免疫系统的许多机理所隐含的丰富思想与工程问题相结合的研究还处于初步阶段。

从应用角度,在人工免疫系统中,已有的大部分基于免疫机理的算法是针对具体问题而设计的,且免疫算子的设计不具有通用性,因而限制了其应用范围。另一方面,由于免疫系统是一种极为复杂的自适应系统,其动态机制的模拟及各种机制的组合是一种复杂过程,因此建立免疫算法的一般框架极为困难,导致至今有各种类型的基于免疫的算法。另外,目前人工免疫系统的研究正处于大量开发智能方法阶段,并将所提出的方法用于具体问题,已有基于免疫的方法大致包括五类:基于免疫的智能优化算法、免疫网络算法、模式跟踪算法、阴性选择算法及免疫 Agent 算法,这些方法的应用越来越广泛。人工免疫系统属于应用型的研究方向,许多基于免疫的方法已在实际问题中获得广泛而有效的应用。但这些方法的理论研究极为罕见,有必要对这些方法的理论进行分析,进一步揭示这些算法的固有属性和搜索行为特性,为工程应用人员提供理论指导。为了对基于免疫的计算智能方法的研究现状有更清楚的认识,以下对这些方法的研究作概述性介绍。

二、基于免疫的智能优化算法现状分析

随着最优化技术的发展,经典的优化技术已很难适合于科技领域中出现的复杂非线性问题,计算智能方法正迎接各领域出现的复杂优化问题的挑战,正在作为人工智能的重要分支在向前快速发展。在此领域中,免疫系统的多样性机理与遗传算法相结合的算法,以及从免疫学原理出发所获得的算法已广泛应用于工程领域,这说明开发免疫系统,构建免疫模型,探讨算法及模型的理论基础和应用已作为人工智能研究的重要问题引起了广泛关注。基于免疫的算法之所以获得广泛重

视,原因在于这种算法为基于群体的随机搜索算法,且能有效克服其他智能算法的早熟现象、群体多样性不足及搜索速度慢等问题,因此探讨免疫系统的运行机理,模拟其运行机制并提出免疫算法解决实际问题具有理论和现实意义。

基于生物免疫的算法研究起源于 20 世纪 90 年代初期,在这一阶段, Forrest 等将免疫系统中抗体识别抗原的机理与遗传算法 GA 相结合提出了免疫遗传算法^[10]。之后又出现了许多基于免疫的算法,例如免疫遗传算法^[14~20]、免疫规划算法^[35]、克隆选择算法^[39~42]、模式跟踪算法^[43]等。特别地, Fukuda 等人已建立了公认的免疫遗传算法一般框架^[17],这一算法体现了免疫系统的免疫记忆机理及抗体的促进与抑制机制。其次,从免疫学原理出发的一般免疫算法框架尚未建立,目前仅有 de Castro 提出的克隆选择算法及 Gaspar A 等提出的模式跟踪算法具有代表性,但这些算法都仅从免疫系统的特定侧面提出,如克隆选择算法模拟了克隆选择原理的简化机制。这一算法对低维优化问题及城市数较少的 TSP 有一定的有效性,但也存在不少问题。例如,未体现抗体之间的相互作用关系,算法的收敛性难以保证,且在处理高维的复杂多模态函数优化问题时,算法搜索性能不理想。总之,深入挖掘复杂免疫系统的内在机制,提取有益于构建免疫算法的免疫特征是目前人工免疫系统理论及应用领域的重要研究课题。开发一般性免疫算法框架的研究仍然处于不断探索之中,但无论这种算法的框架如何建立,它应满足如下四个条件:(1)既然属于仿生算法,因此它应尽可能拟合免疫系统的进化过程;(2)结构尽可能简单并具有一般性,且便于用数学方法进行理论探讨;(3)容易与工程问题结合,且获得的效果尽可能理想;(4)与其他智能算法易于结合。

(一) 免疫遗传算法发展概况

在此将免疫机理与遗传算法结合建立的算法称为免疫遗传算法(IGA)。免疫系统中的群体多样性机理可改进 GA 的多样性,文献^[10]利用抗体识别抗原的特征,建立了抗原与抗体匹配的评价规则,并将免疫系统多样性机理通过适应度算法刻画(即进化群体中个体的适应度评价方案)。这种适应度算法本质上是共享适应度法,不过在此回避了确定小生境半径的困难。文献^[10~13]在这一适应度算法基础上,利用 GA 进化抗体群,提出了基于遗传算法的免疫系统模型,并用于模式识别及多模态函数优化问题。这种免疫系统模型本质上是利用免疫系统中抗体识别抗原的机制改善 GA 的局部搜索性能,这标志着免疫遗传算法的最早问世。另外这种适应度算法被文献^[26,27,64,65]成功地用于处理约束优化问题中的约束条件。特别地,文献^[65]将此算法与 GA 结合,将多目标约束优化问题转换为基于权重系数的单目标无约束优化问题,通过选择多组权重向量获得多个单目标优化问题,依据这些单目标优化问题,复制抗体群中的部分抗体构成抗原群,利用抗体适应度算法及 GA 进化抗体,最终获得优化问题的多个 Pareto 最优解。此算法已被用于实际多目标优化设计,同时此算法的出现标志着人工免疫系统已步入实际多目标优

化问题的解决之中。

免疫系统的多样性思想应用于 GA 后,相继出现了许多新的方法设计抗体适应度或抗体选择概率方案,进而获得多种类型的免疫遗传算法。以下对几种具有代表性的免疫遗传算法作扼要介绍。

文献 [14] 依据抗体浓度及抗体与抗原的亲合力,设计了抗体的期望繁殖率,并以此期望繁殖率作为抗体的适应度,然后利用 GA 进化抗体群,最终获得所需答案。文献 [67] 将文献 [14] 的算法与其他智能算法进行比较分析获得这种算法的有效性。文献 [15~17,25] 利用文献 [14] 的方法设计期望繁殖率,依据这种期望繁殖率确定性地清除部分低期望繁殖率的抗体,剩下的抗体参与交叉与突变,最终获得所需的解。文献 [25] 与文献 [15~17] 的主要区别是,在利用设计的抗体选择概率选择抗体之前,需要利用设计的两种免疫记忆算子对抗体群进行处理。免疫记忆算子对好的抗体的记忆,对浓度高的抗体抑制性记忆。这种算法使好的抗体参与选择,利用局部搜索或随机产生新抗体的方法取代浓度较高的抗体。这种方法的主要优点是 GA 的选择所作用的抗体群的多样性较好,搜索效果较明显。但此算法仅以 12 座城市为例说明此算法的搜索性能,对于城市数较大的情形未给予仿真结果。

文献 [18~19] 将免疫系统的特异性免疫特征与遗传算法相结合提出了一种独特的免疫遗传算法。这种算法的思想是首先设计特定的免疫算子,然后将这算子与 GA 的交叉和突变算子结合并对当前群体进化,免疫算子由接种疫苗及免疫选择构成。接种疫苗是在问题的先验知识指导下,对进化群体中部分个体的基因进行修改。免疫选择是在适应度改进的个体和适应度未改进的个体的父代个体所构成的群体中,利用模拟退火选择算子选择个体构成下一群体。这种算法首先需依据问题的特征信息提取疫苗,即设定特定的基因位上的基因,算法在执行过程中,个体依据事先获得的疫苗信息提取疫苗,这有助于提高个体的适应度。这种算法主要是针对规模与难易程度成正比的优化问题设计的,特别是 TSP。但是这种算法对于处理多峰值函数或特征信息提取较困难的优化问题时,其优越性不明显。鉴于此算法的特征信息提取困难,文献 [20] 对这种算法作了适当改进,即利用进化群体中最好的和次好的个体作为疫苗,克服了文献 [18] 的特征信息提取的困难。同时接种疫苗算子作用于进化群体时,对于被接种的个体来说,其疫苗的基因数是被动态确定的,这增强了算法的自适应能力。改进后的算法在搜索性能和实用范围上明显优于文献 [18] 的算法,但由于接种疫苗选择的特殊性,实际应用时,不可避免地会仅获得优化问题的次优解或局部最优解。

文献 [29~30] 利用信息熵理论设计抗体浓度,并依据抗体浓度和抗体适应度概率设计抗体的选择概率,进而通过利用 GA 进化抗体群获得最终的解,所获得的算法已有效地应用于 TSP 和装箱问题。

综上所述,免疫遗传算法的种类较多,相对来说,这类算法的应用研究较多。这些算法的主要任务是设计特定的增强群体多样性的免疫算子并与 GA 结合,改

进 GA 的搜索性能。这进一步说明 GA 中的选择算子是导致算法陷入局部搜索的关键因素之一,因此遗传算法与免疫机理的结合能较大幅度地改善 GA 的搜索性能。大量的实例表明免疫遗传算法比基本遗传算法能更有效地处理优化问题。

(二) 基于免疫学原理的优化算法

已见报道的基于免疫学原理的算法较少,目前仅有克隆选择算法、免疫网络算法、模式跟踪算法、免疫 Agent 算法及阴性选择算法等五种类型。

能合理解释免疫系统如何清除外在物质(抗原)的原理包括克隆选择原理及独特型免疫网络原理,这些理论的有机结合刻画了免疫系统中抗体与抗原的作用关系,阐述了抗体如何学习抗原模式的过程。抗体学习本身是一种进化过程,因此依据这些理论可开发智能方法解决实际优化问题。这种想法已被 de Castro^[39]于 2000 年实现。文献 [39, 40] 成功地建立了基于二进制编码的克隆选择算法,这种算法的免疫学理论基础是克隆选择原理,算法的设计思想是: (1) 随机产生初始群体; (2) 选择亲和力最高的抗体繁殖一定数目的克隆并进行突变; (3) 突变的克隆若其亲和力比母体高,则取代母体,否则被清除; (4) 随机选择一定数目的抗体取代亲和力较低的抗体。克隆选择算法的出现标志着从免疫学角度开发的第一种智能算法的出现,尽管此算法存在着对特定的优化问题效果不理想的现象,但它的出现标志着从免疫系统自身机理出发开发智能优化算法解决优化问题的开始。

文献 [39] 的算法出现后,文献 [41] 在此算法基础上,仍然从克隆选择原理出发,提出了基于实数编码的广义克隆选择算法。该算法与文献 [39] 的区别在于: (1) 文献 [39] 仅要求亲和力最高的抗体进行克隆,而文献 [41] 却要求进化群体中所有的抗体都各自繁殖一定数目克隆; (2) 文献 [39] 的克隆按二进制编码进行单点突变,而文献 [41] 的克隆按实数编码进行高斯变异。文献 [42] 与文献 [39] 的思想方法很类似,只不过这两文献的关键区别是克隆细胞的突变规则,文献 [42] 采用的是多字符的编码规则。

从克隆选择原理出发所获得的克隆选择算法能很好地解释抗体应答抗原的部分作用机制。但这种算法未体现抗体之间的作用关系,因而导致算法搜索过程中出现群体多样性不足的现象。然而独特型免疫网络原理刻画了免疫系统中抗体与抗体、抗体与抗原之间的作用关系,这种关系表现为调节机制,其中抗体之间的调节表现为抗体之间的抑制和促进,即亲和力高且浓度低或期望繁殖率高的抗体受到鼓励或促进,反之则受到抑制。抗体的抑制和促进可根据抗体的浓度及亲和力设计随机选择算子进行刻画。

文献 [55] 在总结了文献 [39] 的不足之后,将抗体的抑制机制与克隆选择算法结合,建立了基于实数编码的人工免疫网络模型。此模型覆盖了文献 [39 ~ 42] 的主要思想,具有一定的通用性,算法设计中主要是将克隆抑制这一确定性操作与克隆选择算法相结合。但由于在此算法中,包含抗体繁殖的克隆群体经过突变后的克

隆群的平均适应度仅在高于母体群的平均适应度时才进入下一操作,这隐含了内循环的计算量加大,同时当出现突变克隆群的平均适应度几乎不可能高于母体群的平均适应度时,算法将出现无限循环.因此有待进一步考虑提出合理且有效的一般性人工免疫系统优化模型.

另外,基于免疫学原理、进化算法或数学规划方法的多目标优化算法^[62~66,68~81]不断出现,特别文献[62]基于克隆选择原理提出了一种多目标免疫系统算法,用于解决含约束或无约束的多目标优化问题.该算法的思想是在算法之外设置一记忆集合,将进化群体中最好的个体复制于记忆集合,且该集合不断被更新.然后将进化群体划分为抗体群和抗原群,依据抗原与抗体的匹配度选择抗体进行克隆及突变,母体与突变克隆集共同确定新群体.此算法体现了 Pareto 最优解的概念及并行处理思想,同时利用记忆细胞集合搜集进化群体中最好个体.但该算法存在着对约束优化问题处理效果不佳的不足,进而文献[66]对此算法作了改进,改进后的算法对约束条件较少及非线性约束性不强的优化问题有一定效果.文献[62]的出现体现了以免疫学原理为依据开发免疫算法解决多目标优化问题具有较大潜力.

从以上分析可看出,克隆选择原理及独特型免疫网络原理的结合更能合理地反映免疫系统与抗原的作用机制.文献[55]仅将抗体的静态抑制思想与克隆选择算法结合,但未引入抗体的促进和抑制的动态调节思想.这导致群体多样性的缺乏及抗体的选择不具有随机性,仅有适应度改善的抗体或克隆被保存,这隐含着算法出现搜索陷入局部区域的可能.因此我们认为抗体或克隆的选择应具有随机性,同时开发既能合理体现免疫系统应答抗原的原理及机理,又能有效解决工程问题的智能优化算法.应该不仅包含克隆选择原理及克隆抑制机制,而且包括抗体之间的抑制和促进的思想及动态平衡维持所应具备的其他机制,这是我们将考虑的问题之一.

(三) 基于免疫原理的其他算法概述

文献[43]基于克隆选择原理及记忆细胞再利用功能,提出了基于二进制编码的模式跟踪算法,该算法的出现表明克隆选择原理在工程问题中形成了又一应用分支.此算法的设计思想是将抗原视为当前群体中的最好抗体,对当前群体设置两种激励度,利用其中一种激励度选择此群体中较好的抗体进行克隆及单点突变,对母体及亲和力比母体高的突变克隆构成的群体作用 K 联赛选择.同时依据抗体的另一激励度并用 K 联赛选择对未被激活的抗体进行选择,这两种选择所获得的个体构成新抗体群.该算法与克隆选择算法相似,但此算法利用随机选择的方式促使进化群体避免陷入局部最优区域搜索.事例仿真说明这种算法对于模式跟踪问题是行之有效的,同时此算法可用于网络检测问题.

文献[44]基于免疫系统的免疫特征,通过将抗体作为信息处理的识别器(或 Agent),利用 Agent 结构和网络结构建立了一种免疫 Agent 算法,并用于含噪声的

定常时不变系统的噪声自适应控制。此算法由多样性识别器的产生、自体耐受的建立及非自我的记忆组成,依据系统的输出误差对识别器的参数进化,进而识别器变为抑制噪声影响系统行为的中和器。这种算法的特点是:(1)体现了记忆和自我与非自我的识别特征;(2)抗体作为识别器,各识别器的参数调整及识别器之间的交互通过系统的输出误差完成;(3)具有网络结构和 Agent 特征,各 Agent 通过局部记忆产生进化机制。

文献 [45] 利用免疫系统中负选择 (negative selection) 的思想设计了一种阴性选择算法,它被有效地应用于入侵检测和工具爆破问题^[46]。这种算法由定义自我、产生探测器及监督异常发生构成,其中自我细胞是指正常模式(若对于计算机检测,则指计算机正常情况下的模式结构),定义自我是指确定自我细胞,产生探测器是指随机产生大量的抗体,监督异常发生是指随机产生的抗体若与自我细胞匹配则被清除,否则将作为监测器。但由于此算法的抗体的产生完全处于随机状态,因此此算法的计算复杂度是相当高的。随后,文献 [47] 将文献 [10] 的适应度算法与文献 [45] 的阴性选择算法结合提出改进的阴性选择算法。在此算法中,抗原群由已知的大量正常模式结构组成,同时 GA 被用于进化监测器的模式结构,最后则可获得最佳的监测器。

文献 [63] 在进化规划算法中引入辞典排序和 Pareto 选择策略,并利用免疫系统的自我与非自我的识别特征处理计算机入侵检测问题。该方法属于多目标优化方法,但重点在于应用,而不在于算法探讨。文献 [35] 在进化规划算法中引入与文献 [18] 相同的免疫算子获得免疫规划算法。这两种算法的出现标志着免疫系统的机理与进化规划算法的结合。

以上具体介绍了从免疫系统的原理或免疫特征出发所获得的几种算法,这些算法从各自的角度提供了人工免疫系统的研究方向。

三、免疫网络算法发展回顾

自 Jerne 提出独特型免疫网络之后,以免疫网络理论为基础所建立的人工免疫网络模型已广泛应用于工程领域,并且免疫网络与其他智能技术的结合已成为一种研究发展趋势。如文献 [51, 52] 开发了一种 B 细胞网络模型,获得了相应的学习算法,并成功用于 DNA 识别。文献 [53] 针对用户对网站需求、网络数据噪声处理及网络个性化的迫切需求,提出了一种人工模糊免疫系统解决数据挖掘问题。此文在文献 [51] 的基础上,将抗体与抗原及抗体与抗体的结合程度(即亲和力)利用模糊隶属度描述,借助于此隶属度设计抗体的激励度,其他设计与文献 [51] 相似。此算法极大改进了文献 [51] 的免疫网络,并成功地用于网络挖掘问题。文献 [54] 从独特型免疫网络原理出发开发了一种基于实数编码的人工免疫网络模型并用于数据压缩,此模型设计的关键在于克隆选择原理的应用,同时此模型的特点在于算法

的群体规模动态调节。

文献 [56] 利用免疫网络中的动态微分方程建立了类似于 BP 网络的递推免疫网络算法, 此算法已用于调整 PID 系统的参数。其优点在于: (1) 免疫网络与神经网络的结合对工程控制问题的解决有较大益处; (2) 体现了抗体的抑制和促进的思想、网络的自组织能力及网络结构的不确定性等特点。文献 [36~38, 57, 58] 利用抗体的促进与抑制机制和 GA 结合构建免疫遗传算法调整神经网络的权值。此类算法的相似之处在于对抗体群中浓度较高的抗体用随机产生的新抗体取代, 然后将交叉和突变算子作用于抗体群。这种算法与神经网络的衔接是将网络中的权值变量构成抗体, 利用误差确定目标函数, 进而利用免疫遗传算法可获得网络权值。算法的优点在于免疫系统的动态特征与神经网络的结合, 克服了神经网络陷入局部搜索的不足。文献 [59] 基于 B 细胞网络理论建立了有别于文献 [51] 的数据分析算法, 同时文献 [60] 将此算法与单链聚类算法 (single linkage clustering) 和自组织神经网络算法进行比较获得此算法在数据挖掘方面的有效性。通过此文的比较获知, 这种算法具有能有效发现未知模式、对样本的分类较详细, 且能非监督地进行聚类等优点。

四、人工免疫系统原理概述

受到免疫系统与外在环境的作用机制及免疫学中基本概念的思想启发, 开发智能方法解决工程问题已成为人工智能领域中出现的新研究方向。国内外不少学者致力于将免疫系统的特征和原理与实际问题的结合, 同时有不少研究成果在工程问题中获得了广泛应用。以下介绍免疫学中克隆选择原理、独特型免疫网络原理及 T 细胞免疫调节原理与人工免疫系统相衔接的部分机制, 这些机制在免疫算法构建中发挥着重要的作用。特别强调的是, 由于 B 细胞表面仅携带一种类型的抗体, 因此以后的叙述中将抗体和 B 细胞不加区别。

(1) 抗体与抗原: 体液免疫过程主要是抗体与抗原的相互作用。抗原作为被处理对象, 抗体作为学习抗原的主体, 抗体如何学习抗原并成为研究的主要问题。借助于抗体学习抗原的思想, 可将要解决的工程问题或问题的答案视为抗原, 需获的候选答案视为抗体, 模拟抗体学习抗原的机制可寻求所处理问题的答案。例如, 在免疫网络聚类算法中, 抗体视为进化的候选聚类点, 抗原视为已知的样本数据。

(2) 克隆选择: 克隆选择是一种确定性的选择方式, 这种方式引导识别抗原能力强的抗体参与免疫应答, 提高识别抗原的能力。该思想反映在免疫算法上则是进化群 (候选解群) 中较高亲和力的抗体 (候选解) 被选择学习抗原, 即进化群体中较好的解参与进化。

(3) 细胞克隆及记忆细胞获取: 此机理的主要作用在于分化免疫细胞。分化的细胞中, 一部分细胞演变为记忆细胞, 另一部分则繁殖大量的无性繁殖系 (即克隆),

这些克隆参与进化并演变为浆细胞。这种机理反映在算法上,记忆细胞则为算法进化到当前群体时,所获得的较好解,这些解又逐渐被更好的解取代。记忆细胞的作用在于为相同或相似问题的解决提供初始个体,有助于提高算法搜索最优解的性能。免疫细胞所繁殖的克隆(即问题的候选解)的演变提借助了算法探测更好解的机会,起到增强算法局部搜索能力的作用。

(4) 亲和成熟:通过对克隆细胞的基因重复进行亲和突变完成。其目的在于加速学习抗原的进程。其反映在算法上则是克隆细胞通过变异产生较好的候选解,以至于逐步获最优解。

(5) 克隆抑制:对突变后的克隆群进行处理,相似的或亲和力低的克隆被消除,而高亲和力的克隆被保存。该思想反映在算法上则为进化群体被划分为若干个独立的子群,每个子群中相似度高且亲和力低的克隆被确定性地清除,这种机制有助于减轻母体群和突变的克隆群构成的混合群体的选择压力。

(6) 动态平衡维持:这种机制是自我抗体参与并维持免疫系统动力学特性的主要环节,即它保存一定数量较好和较差的抗体,同时调节抗体群的规模。此思想反映在算法上则表现为按照特定的随机方式选择好、中、差抗体,并插入一定数量的自我抗体,调节被选中的抗体构成的群体的规模及多样性。

(7) T 细胞调节: T_h 、 T_s 细胞对 B 细胞的应答具有调节作用, T_h 、 T_s 细胞被抗原激活的程度反映了 B 细胞被激活的程度。T 细胞对 B 细胞的调节可被用于反馈控制系统中设计控制器,即将实时系统的输出误差视为抗原,利用 T 细胞调节原理设计控制器。此控制器的输出作为实时系统的输入,通过控制系统的不断作用,实时系统的输出误差逐渐趋于 0。

第三节 本篇研究的主要内容及意义

免疫算法理论及应用作为此领域的重要分支,已受到国内外计算智能学者的密切重视。目前基于免疫机理的优化算法较多,但以免疫学原理为依据的算法较少。已有算法的宗旨是力图建立免疫算法的一般框架,但比较困难。主要原因有几点:(1) 免疫系统作为自我防御系统,其本身的动力学行为特性极为复杂,如何挖掘其动力学机制,结合免疫系统的防御功能,建立一种既能充分反映免疫系统与外在环境发生作用的行为特性,又能有效地解决现实问题的免疫算法框架是一项艰巨的任务;(2) 免疫算法的免疫学理论的选择以及如何利用数学工具建立免疫算法的数学基础较难;(3) 相关的智能算法的理论探讨甚少,可借鉴的智能算法的理论结果较少;(4) 免疫算子的数学定义及免疫算法与工程问题的衔接关系如何搭建较难。这些问题是在人工免疫系统理论及应用作为新的研究领域不断被深入研究的前提下产生的。其次,免疫网络理论作为免疫学中核心内容之一,这些理论对研究人员开发免疫网络算法处理工程问题起到了抛砖引玉的作用,尤其是免疫网络理

论与神经网络理论的结合是开发免疫网络的主要动力。鉴于免疫算法及免疫网络是人工免疫系统研究的核心内容之一,本部分主要研究如下内容。

一、理论方面

(1) 基于体液免疫理论,建立一般性免疫算法的理论及应用框架,给出免疫算法的概念、原理、算法描述、稳定性定义、稳定性结论、收敛性、收敛速度估计及鲁棒性。

(2) 提出形态空间上几种变形免疫算法,即小生境免疫算法、动态规模免疫算法、模糊控制免疫算法、约束优化免疫算法,并进行原理、收敛性、搜索性能及应用分析,同时与其他智能算法比较。这些算法是针对非约束函数优化及约束函数优化进行设计的,同时从不同侧面反映了免疫系统的动力学特征。

(3) 提出多目标优化免疫算法(约束、无约束),给予算法描述、与其他智能算法的比较及算法收敛性。

(4) 提出模糊免疫网络分类算法,介绍其原理及算子设计,并进行性能测试等。

二、应用方面

(1) 将一般免疫算法、变形免疫算法(约束、非约束)用于函数优化问题进行性能测试及比较。一般免疫算法被用于 TSP 及系统辨识,变形免疫算法中小生境免疫算法被用于磁盘驱动器的参数辨识,动态规模免疫算法被用于极点配置问题,约束优化免疫算法被用于气压钢优化设计。

(2) 非约束多目标优化免疫算法被用于机械锻锤优化设计,约束多目标优化免疫算法被用于车床加工设计。

(3) 模糊免疫网络聚类算法用于数据分类问题。

以上两方面研究的目的在于从免疫学的角度建立解决一般最优化问题的免疫算法,并建立免疫算法的自身理论体系,探讨其应用前景以及深化免疫机制及丰富和完善免疫算法的内涵,这对人工免疫系统理论及应用领域中的重要分支(即免疫算法理论及应用)无论是理论上还是应用上都有一定的价值。其理论价值体现在不仅为免疫算法的应用提供可靠保证,而且为探讨免疫算法及相关智能算法的理论提供新的思路,同时为免疫算法的深入研究奠定理论基础。其实际意义在于提供用新的智能方法处理工程问题的工具,并可将所提出的算法直接作用于实际的优化对象,克服了用数学规划方法处理非线性优化问题的困难,特别是非线性约束条件下的非线性优化问题(单目标、多目标)。

第四节 最优化问题及分类

最优化问题可分为组合优化及函数优化两大类, 这两类问题皆包含约束优化及非约束优化. 为便于引用, 以下分别对各类优化问题进行描述.

一、基于二进制编码的优化问题

设 X 表示由 $0, 1$ 构成的字符集, 所有长度为 l 的字符串构成的集合为 $S = \{x | x \in X^l\}$, $f: S \rightarrow \mathbb{R}$ 为目标函数, $g_i: S \rightarrow \mathbb{R}, h_j: S \rightarrow \mathbb{R}, i = 1, 2, \dots, L, j = 1, 2, \dots, M$ 皆为约束函数, 则基于二进制编码的非约束极小化问题可描述为

$$\text{NBP} \quad \min_{x \in S} f(x),$$

具有约束条件的极小化问题描述为

$$\begin{aligned} \text{CBP} \quad & \min_{x \in S} f(x), \\ & h_i(x) = 0, g_j(x) \geq 0, i = 1, 2, \dots, L, j = 1, 2, \dots, M. \end{aligned}$$

二、多字符集上的优化问题

设 X 表示由 n 个不同的字符构成的字符集, 个体长度为 n , 每一个个体为 X 中 n 个不同字符构成的一个排序, S 表示由所有这样的个体构成的集合. 目标函数及约束条件与基于二进制编码的优化问题中描述相似, 则相应的非约束极小化问题记为 NCP, 约束极小化问题记为 CCP.

三、函数优化问题

设 \mathbb{R}^p 为 p 维欧氏空间, $D \subset \mathbb{R}^p, f: D \rightarrow \mathbb{R}^m$ 为目标函数或目标向量, $h_i(x), g_j(x)$ 皆为 D 上的约束函数, $1 \leq i \leq L, 1 \leq j \leq M$. 当 $m = 1$ 时, 对应于 NBP 的问题称为单目标非约束函数优化问题, 记为 NFP, 对应于 CBP 的问题称为单目标约束函数优化问题, 记为 CFP, 其中 D 可为无界区域; 当 $m > 1$ 时, 对 D 的限制与 $m = 1$ 时的情形相同, 非约束多目标极小化问题可描述为

$$\text{NCMO} \quad \min_{x \in D} f(x) = (f_1(x), f_2(x), \dots, f_m(x)),$$

含约束极小化问题可描述为

$$\begin{aligned} \text{CMO} \quad & \min_{x \in D} f(x) = (f_1(x), f_2(x), \dots, f_m(x)), \\ & h_i(x) = 0, g_j(x) \leq 0, i = 1, 2, \dots, L, j = 1, 2, \dots, M. \end{aligned}$$

第五节 测试问题及性质分析

例 2.5.1 TSP

TSP 描述为给定 n 个城市和各城市之间的距离, 寻找一条最短路径, 使访问者经过所有城市并且每一城市仅被访问一次, 最终返回出发点. 以下将此问题建立数学模型.

设有 n 个城市 A_1, A_2, \dots, A_n , 城市 A_i, A_j 之间的距离表示为 d_{ij} . 这 n 个城市的所有可能的排列构成的集合表示为 $P = \{A_{i_1} A_{i_2} \dots A_{i_n} \mid i_l \neq i_m, 1 \leq i_k, m, n \leq n\}$, $|P| = n!$, 则 $A_{i_1} A_{i_2} \dots A_{i_n} A_{i_1}$ 构成一条访问 n 个城市的路径, 用 $d(A_{i_1}, A_{i_2}, \dots, A_{i_n})$ 表示在这条路径上相邻及首尾城市之间的距离之和, 即

$$d(A_{i_1}, A_{i_2}, \dots, A_{i_n}) = \sum_{k=1}^{n-1} d_{i_k i_{k+1}} + d_{i_n i_1}.$$

于是含 n 个城市的 TSP 的数学模型可描述为

$$\min_{A_{i_1} A_{i_2} \dots A_{i_n} \in P} d(A_{i_1}, A_{i_2}, \dots, A_{i_n}).$$

例 2.5.2 多模态函数及性质分析

为便于引用, 以下列举几种公用的测试函数^[79,80], 并对其性质作分析

$$F_1: f(x, y) = 100(x^2 - y)^2 + (1 - x)^2, -2.048 \leq x, y \leq 2.048;$$

$$F_2: f(x, y) = 0.5 + \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{[1.0 + 0.001(x^2 + y^2)]^2}, -100 \leq x, y \leq 100;$$

$$F_3: f(x, y) = \sum_{i=1}^5 i \cos[(i+1)x + i] \cdot \sum_{i=1}^5 i \cos[(i+1)y + i], -10 \leq x, y \leq 10;$$

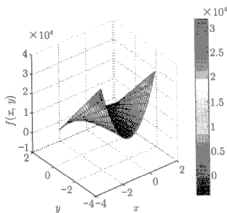
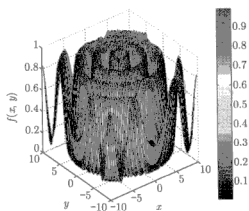
$$F_4: f(x, y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2, -3 \leq x \leq 3, -2 \leq y \leq 2.$$

所列举的函数皆为非凸函数, 其中 F_1 为单峰值的病态函数, F_2 、 F_3 及 F_4 为多峰值函数, 通常这些函数用作测试智能算法优越性的基础. 这些函数的全局最优优点要么在已知区域的特殊位置, 要么出现在区域的边界位置, 而且在最优解附近有许多局部最优解干扰算法搜索, 对于一般的智能算法搜索较困难.

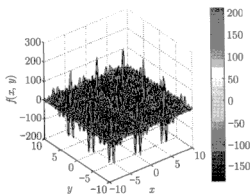
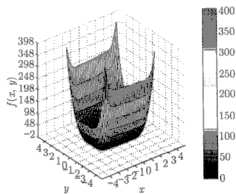
函数 F_1 的图形如图 2-5 所示, 该函数仅在已知区域的端点 $(-2.048, -2.048)$ 处取最大值 3905.93, 在点 $(2.048, -2.048)$ 处取次优值 3897.73. 同时有唯一取最小值 0 的点 $(1, 1)$, 此点为已知区域的内点, 并且由二元可微函数取最值的必要条件易于获得该点, 但在该点附近却有大量的局部最优优点. 另外该函数的最大值和次优值尽管相差不大, 但取最大值的点和取次优值的点相距较远, 一般的智能算法仅能搜索到次优解.

从图 2-6 可以看出, 函数 F_2 是极为困难的多峰值函数, 其包含有无穷多个局部最优解和全局最优解. 该函数的最大值为 0.997493, 最小值为 0, 取最大值的最优

解满足 $x^2 + y^2 = 2.44$. 特别有 8 个最优解为 $(\pm 1, \pm 1.2)$ 及 $(\pm 1.2, \pm 1)$, 另外 F_2 仅在 $(0,0)$ 处取最小值. 尽管该函数仅含一个取最小值的解, 但在此解附近有无穷多个取局部极小值的解, 从而算法搜索极为困难.

图 2-5 函数 F_1 图 2-6 函数 F_2

函数 F_3 有大量的局部最优解 (如图 2-7), 最大值约为 210.482, 最小值约为 -186.73. 此函数存在较多的全局最优解, 这些解分布相对均匀, 在全局最优解附近包含有较多的局部最优解, 并且局部最优解与全局最优解相互交叉排列.

图 2-7 函数 F_3 图 2-8 函数 F_4

函数 F_4 有取最小值的两个点 $(-3, -2)$ 及 $(3, 2)$, 此两点出现在已知区域的端点处 (如图 2-8), 同时在已知区域内包含大量的取局部极小值的点及多个取最小值的点, 最小值约为 -1.031628. 由于在取最小值的点附近有大量的局部最优解, 因此给算法寻优带来困难, 特别对于搜索最大值的点, 一般的智能算法较难完成.

例 2.5.3 约束优化问题

约束优化问题是一种极为困难的优化问题, 这种问题的关键之一在于约束条

件的处理, 特别当约束条件较多及优化问题的维数较高时, 处理比较困难. 在此列举两个约束优化问题^[81]用于测试所构建的免疫算法搜索性能.

问题 1

$$\begin{aligned} \min f_0(x_1, x_2) &= (x_1^2 + x_2 - 1)^2 + (x_1 + x_2^2 - 7)^2, \\ \text{s.t. } f_1(x_1, x_2) &= 4.84 - x_1^2 - (x_2 - 2.5)^2 \geq 0, \\ f_2(x_1, x_2) &= (x_1 - 0.05)^2 + (x_2 - 2.5)^2 - 4.48 \geq 0, \\ 0 \leq x_1 \leq 6, 0 \leq x_2 \leq 6. \end{aligned}$$

从图 2-9 可知, 该问题中满足约束条件的点已集中在已知区域的较窄部分, 即在点 (1.68974, 1.15096) 附近的区域, 因而对一般的算法搜索最优解有一定的难度.

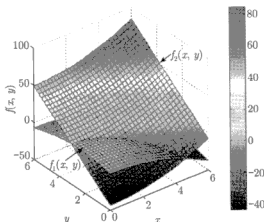


图 2-9 问题 1 的函数 $f_1(x, y)$ 及 $f_2(x, y)$

问题 2

$$\begin{aligned} \min f_0(X) &= x_1 + x_2 + x_3, \\ \text{s.t. } f_1(X) &= 1 - 0.0025(x_4 + x_6) \geq 0, f_2(X) = 1 - 0.0025(x_5 + x_7 - x_4) \geq 0, \\ f_3(X) &= 1 - 0.01(x_8 - x_5) \geq 0, f_4(X) = x_1x_6 - 833.33252x_4 + 100x_1 + 83333.333 \geq 0, \\ f_5(X) &= x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0, \\ f_6(X) &= x_3x_8 - x_3x_5 + 2500x_5 - 1250000 \geq 0, \\ 100 \leq x_1 \leq 10000, 1000 \leq x_2, x_3 \leq 10000, 10 \leq x_i \leq 1000, i &= 4, 5, \dots, 8. \end{aligned}$$

问题 2 是一种极为困难的优化问题, 其突出特点是维数较高且含多个约束条件, 同时约束条件之间相互制约力较强, 而且满足约束的有效区域的范围相当窄. 特别地, 约束条件是制约算法搜索困难的关键因素.

第六节 本章小结

免疫学的概念、机理、原理是开发人工免疫系统的生物理论依据,本章首先对免疫学的基本概念、免疫应答的运行机制及免疫细胞的主要免疫功能进行了概述性介绍和分析,对自适应免疫应答所包含的两种应答(细胞免疫及体液免疫)的运行机制进行理论阐述和概括,分析各机理的作用,给予免疫系统多样性特征描述,概括免疫系统的主要特征,给出了免疫学基本概念、原理与工程问题相衔接的人工免疫系统基本理论,这些论述为本部分开发人工免疫系统创造了必要条件,进而分析了人工免疫系统中智能优化算法及免疫网络算法的发展现状及关注的问题,明确本部分研究的主要内容,最后给出和分析了本部分所需引用的测试问题。

参 考 文 献

- [1] 龚非力. 医学免疫学. 北京: 科学出版社, 2003. 176
- [2] de Castro L N, Von Zuben F J. Artificial Immune Systems: Part I-Basic Theory and Applications. Technical Report, TR-DCA 01/99, 1999. 95
- [3] Dasgupta D. Artificial Immune Systems and Their Applications. New York: Springer-Verlag, 1999. 300
- [4] de Castro L N, Von Zuben F J. Artificial Immune Systems: Part II-A Survey of Applications, Tech., Rep-R T DCA, 2000. 65
- [5] 莫宏伟. 人工免疫系统原理与应用. 哈尔滨: 哈尔滨工业大学出版社, 2002. 66
- [6] 丁永生, 任立红. 人工免疫系统: 理论与应用. 模式识别与人工智能, 2000, 13(1): 52~59
- [7] 肖人彬, 王磊. 人工免疫系统: 原理、模型、分析及展望. 计算机学报, 25(12): 1283~1293
- [8] 韦巍, 张国宏. 人工免疫系统及其在控制系统中的应用. 控制理论与应用, 2002, 19(2): 157~160
- [9] Farmer J D, Packard N H, Perelson A S. The Immune System, Adaptation, and Machine Learning. Physica D, 1986, 22: 187~204
- [10] Forrest S, Perelson A S. Genetic algorithms and the immune system. In Hans-Paul Schwefel and R. Maenner, editors. Parallel Problem Solving from Nature, Lecture Notes in Computer Science. Berlin: Springer-Verlag, 1991. 320~325
- [11] Forrest S, Javornik B, et al. Using genetic algorithms to explore pattern recognition in the immune system. Evolutionary Computation, 1993, 1(3): 191~211
- [12] Smith R E, Forrest S, Perelson A S. Searching for Diverse, Cooperative Populations. Evolutionary Computation, 1993, 1(2): 127 ~ 149
- [13] Smith R E, Forrest S, Perelson A S. Population Diversity in an Immune System Model: Implications for Genetic Search. In L. Darrell Whitley, editor. Foundations of Genetic Algorithms. California: Morgan Kaufmann Publishers, 1993. 153~165

- [14] Mori K, Tsukiya M, Fukuda T. Immune Algorithm with Searching Diversity and its Application to Resource Allocation Problem. T. IEE Japan, 1993, 113-C (10): 872~878
- [15] Chung J S, Kim M K, Jung H K. Shape Optimization of electromagnetic devices using Immune Algorithm. IEEE Transactions on Magnetics, 1997, 33(2): 1876~1879
- [16] Huang S J. An Immune-Based Optimization Method to Capacitor Placement in a Radial Distribution System. IEEE Transactions on Power Delivery, 2000, 15(2): 744~749
- [17] Fukuda T, Mori K, Tsukiya M. Parallel Search for Multi-modal Function Optimization with Diversity and Learning of Immune Algorithm. In (Ed.) D. Dasgupta. Artificial Immune Systems and Their Applications. New York: Springer-Verlag, 1999. 300
- [18] Jiao Licheng, Wang Lei. A Novel Genetic algorithm Based on Immunity. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2000, 30(5): 552~561
- [19] 王磊, 潘进, 焦李成. 免疫算法. 电子学报, 2000, 28(7): 74~78
- [20] Sun Xiaoyan, Gong Dunwei. Research on a Modified Immunity Genetic Algorithm. Proc. of the 2002 International Conference on Control and Automation, Xiamen, China, Jun. 2002, 777~780
- [21] Opera M, Forrest S. How the immune system generates diversity: Pathogen space converge with random and evolved antibody libraries. Genetic and Evolutionary Computation Conference (GECCO'99), 1999, 2: 1651~1656
- [22] Li Sun, Cai Wensheng, Shao Xue-guang. A Two-dimensional immune algorithm for resolution of overlapping two-way chromatograms. Fresenius J Anal Chem, 2001, 370:16~21
- [23] Shao Xue-guang, Chen Zonghai, Lin Xiangqin. An immune algorithm for resolution of multi-component overlapping chromatograms. Fresenius J. Anal Chem, 2000, 366:10~16
- [24] Huang S J. Application of Immune-Based Optimization Method for Fault-Section Estimation in a Distribution System. IEEE Transactions on power delivery, 2002, 17 (3): 779 ~784
- [25] Endoh S, Toma N, Yamada K. Immune algorithm for n-TSP. Systems, Man, and Cybernetics, 1998, 4(11-14): 3844~3849
- [26] Hajela P, Lee J. Constrained Genetic Search via Schema Adaptation, An Immune Network Solution. In Niels Olhoff and George I. N. Rozvany, editors. Proc. of the First World Congress of Structural and Multidisciplinary Optimization. Germany: Goslar, 1995. 915~920
- [27] Hajela P, Lee J. Constrained Genetic Search via Schema Adaptation: An Immune Network Solution. Structural Optimization, 1996, 12(1): 11~15
- [28] 葛红, 毛宗源. 免疫算法. Proceedings of the 4th word congress on intelligent control and automation, June10-14, 2002, Shanghai, P. R. China, 1784~1788
- [29] 王熙法, 张昱俊, 曹先彬等. 一种基于免疫原理的遗传算法. 小型微型计算机系统, 1999, 20(2):17~20
- [30] 曹先彬, 刘克胜, 王熙法. 基于免疫遗传算法的装箱问题求解. 小型微型计算机系统, 2000, 21(4): 361~363
- [31] 罗小平, 韦巍. 一种基于生物免疫遗传学的冗余机械手轨迹规划新方法. 模式识别与人工智能, 2002, 15(3):199~304

- [32] 汪定伟, 于海斌. 免疫传算法及其在新产品投入计划中的应用. 控制理论与应用, 2002, 19(5):725~730
- [33] 谭志扬. 人工免疫算法在 Flow-shop 问题中的应用. 计算机工程与应用, 2002.14, 98~99
- [34] 曹先彬, 郑振, 刘克胜等. 免疫进化策略及其在二次布局求解中的应用. 计算机工程, 2000, 26(3): 1~10
- [35] 王磊, 潘进, 焦李成. 免疫规划. 计算机学报, 2000, 23(8): 806~812
- [36] 张军, 刘克胜, 王熙法. 一种基于免疫调节和共生进化的神经网络优化设计方法. 计算机研究与发展, 2000, 37(8): 924~930
- [37] 吕岗, 赵鹤鸣, 谭得健. 基于免疫算法的前馈神经网络权值设计. 计算机工程与应用, 2002.17, 31~32
- [38] 曹先彬, 刘克胜, 王熙法. 基于免疫规划的多层前馈网络设计. 软件学报, 1999,10(11):1180~1184
- [39] de Castro L N, Von Zuben F J. The Clonal Selection Algorithm with Engineering Applications. In Workshop Proc. of GECC'00, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, USA, July 2000, 36~37
- [40] de Castro L N, Von Zuben F J. Learning and Optimization Using the Clonal Selection Principle. IEEE Transaction on Evolutionary Computation, 2002, 6(3): 239~251
- [41] Wierzbchoń S T. Function Optimization by the Immune Metaphor. TASK QUARTERLY, 2002, 6(3): 1~16
- [42] Costa A M, Vargas P A, et al. Makespan Minimization on Parallel Processors: An Immune-Based Approach. Evolutionary Computation, CEC '02. Proceedings of the 2002 Congress, 12-17 May 2002, 1: 920~925
- [43] Gaspar A, Collard P. Two Models of Immunization for Time Dependent Optimization. SMC 2000 IEEE International Conference on Systems, Man, and Cybernetics, Nashville, Tennessee, USA, 2000. Special Track on Artificial Immune Systems, 113~118
- [44] Ishida Y, Adachi N. Active noise control by an immune algorithm: adaptation in immune system as an evolution. Proc. of IEEE International Conference on Evolutionary Computation, 20-22 May 1996, 150 ~153
- [45] Forrest S, Hofmeyr S, Somayaji A. Computer Immunology. Communications of the ACM, 1997, 40(10): 88~96
- [46] Dasgupta D, Forrest S. Artificial Immune Systems in Industrial Applications. In Dasgupta D (Ed.). Artificial Immune Systems and Their Applications. Berlin: Springer-Verlag, 1999. 257~268
- [47] Kim J, Bentley P. Negative Selection and Niching by an Artificial Immune System for Network Intrusion Detection. In GECCO-99 Proceedings, 1999, 149~158
- [48] Zhang Zhuhong, Huang Xi-yue and Ma Xiao-xiao. A novel artificial immune algorithm and its application to multi-modal function optimization. Proc. of the International Conference on Control and Automation, Xiamen, China, Jun. 2002, 777~780
- [49] 张著洪, 黄席隼. 一种新的免疫算法及其在多模态函数优化中的应用. 控制理论与应用, 2004, 21(1): 17~21
- [50] 张著洪, 黄席隼. 基于免疫应答原理的免疫算法及其在多模态函数优化中的应用. 重庆大学学报, 2003,

- 26(9): 130~133
- [51] Hunt J E and Cooke D E. An adaptive, distributed learning systems based on the immune system. In Proc. of the IEEE International Conference on Systems, Man and Cybernetics, 1995, 2494~2499
- [52] Hunt J E, Cooke D E. Learning using an artificial immune system. J. of Network and Computer Appl., 1996, 19(4):189~212
- [53] Nasaroui O, Gonzalez F, and Dasgupta D. The Fuzzy Artificial Immune System: Motivations, Basic Concepts, and Application to Clustering and Web Profiling. Proc. of the 2002 IEEE International Conference on Computational Intelligence, 12-17 May 2002, 1: 711~716
- [54] de Castro L N, Von Zuben F J. aiNet: Artificial Immune Network for Data Analysis. In H. A. Abbass H A, Sarker R A, and Newton C S (eds.). Data Mining: A Heuristic Approach. USA: Idea Group Publishing, 2001. 231~259
- [55] de Castro L N, Timmis J. An Artificial Immune network for Multimodal Function Optimization. Evolutionary Computation, 2002-CEC '02, Proc. of the 2002 Congress, 12-17 May 2002, 1: 699~704
- [56] Kim D H. Tuning of a PID Controller Using a Artificial Immune Network Model and Local Fuzzy Set. IFSA World Congress and 20th NAFIPS International Conference, 25-28 July 2001, 15: 2698~2703
- [57] Kim D H. Parameter Tuning of Fuzzy Neural Networks By Immune Algorithm. Proc. of the 2002 IEEE International Conference on Neural Networks and Computational Intelligence, 12-17 May 2002, 1: 408~413
- [58] Kim D H, Lee K Y. Neural Networks Control by Immune Network Algorithm Based Auto-Weight Function Tuning. Proc. of the 2002 International Joint Conference on Neural Networks and Computational Intelligence, 12-17 May 2002. 2: 1469 ~1474
- [59] Timmis J, Neal M, Hunt J. An Artificial Immune System for Data Analysis. Biosystems, 2000. 55(1/3): 143~150
- [60] Timmis J, Neal M, Hunt J. Data Analysis Using Artificial Immune Systems, Cluster Analysis and Kohonen Networks: Some Comparisons. Systems, Man, and Cybernetics, IEEE SMC '99 International Conference, 12-15 Oct. 1999. 3: 922 ~927
- [61] 张著洪, 黄席樾. 基于模糊逻辑和免疫应答原理的免疫网络聚类算法. 见会议文集, 冯浩等编辑. 神经网络与计算智能. 浙江: 浙江大学出版社, 2002. 410
- [62] Carlos A C C and Nareli C C. An Approach to Solve Multiobjective Optimization Problems Based on Artificial Immune System. ICARIS-2002: 1st International Conference on Artificial Immune Systems (ICARIS-2002), University of Kent at Canterbury. 9-11 September 2002, 212~221
- [63] Anchor K P, Zydallis J B, Gunsch G H, et al Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach. In Jonathan Timmis and Peter J. Bentley, editors. First International Conference on Artificial

- Immune Systems (ICARIS'2002). UK: University of Kent at Canterbury, September 2002. 12~21
- [64] Cui Xunxue, Miao Li, Fang Tingjian. Study of Population Diversity of Multiobjective Evolutionary Algorithm Based on Immune and Entropy Principles. In Proceedings of the Congress on Evolutionary Computation (CEC'2001), Piscataway, New Jersey, May 2001, 2: 1316~1321
- [65] Yoo J, Hajela P. Immune network simulations in multicriterion design. *Structural Optimization*, 1999, 18(2/3): 85~94
- [66] Carlos A C C, Nareli C C. Solving Multiobjective Optimization Problems using an Artificial Immune System. Netherlands: Kluwer Academic Publishers, 2002. 36
- [67] Chung J S, Jung H K, Hahn S Y. A Study on Comparison of Optimization Performances between immune Algorithm and other Algorithms. *IEEE Transactions on Magnetics*, 1998, 34(5): 2972~2975
- [68] Osyczka A. Multicriterion Optimization in Engineering with FORTRAN Program. Ellis Horwood, John Wiley, New York, Chichester, Brisbane, Toronto 1984. 301
- [69] Kasprzak E M, Lewis K E. Pareto Analysis in Multiobjective Optimization Using the Colinearity Theorem and Scaling Method. *Structural and Multidisciplinary Optimization*, 2001, 22(3): 208~218
- [70] Leung Yiuwing, Wang Yuping. Multiobjective Programming Using Uniform Design and Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 2000, 30(3): 293~304
- [71] Yoo J, Hajela P. Fuzzy multicriterion design using immune network simulation. *Struct Multi-disc Optim*, Springer-Verlag 2001, 22: 188~197
- [72] 逊崔学. 基于免疫原理的多目标进化算法群体多样性研究. 模式识别与人工智能, 2001, 14(3): 291~295
- [73] 王均. 一种基于多目标的自组织神经网络学习方法. 小型微型计算机系统, 2002, 23(5): 565~568
- [74] 王凌, 郑大钟. 多目标优化的一类模拟退火算法. 计算机工程与应用, 2002, 8, 4~5
- [75] 刘海林, 王宇平, 刘永清. 一种新的正交多目标最优化遗传算法. 计算机工程与应用, 2002, 11, 27~29
- [76] 黄席樾, 张著洪. 基于免疫应答原理的多目标优化算法及其应用. 信息与控制, 2003, 32(3): 209~213
- [77] 张著洪, 黄席樾. 多目标约束优化免疫算法研究及其应用. 模式识别与人工智能, 2003, 16(4): 452~458
- [78] 王凌. 智能优化算法及其应用. 北京: 清华大学出版社 施普林格出版社, 2001. 230
- [79] 周明, 孙树栋. 遗传算法及应用. 北京: 国防工业出版社, 2000. 202
- [80] 张著洪, 黄席樾, 胡小兵. 采用重复交叉操作及最优保留策略的遗传算法. 重庆大学学报, 2002, 25(7): 23~25
- [81] 刘海林, 王宇平, 刘永清. 解约束最优化问题的一个新的多目标进化算法. 计算机工程与应用, 2002, 10, 27~29

第三章 免疫算法理论及应用

第一节 引言

免疫算法是一种模拟免疫系统智能行为的仿生算法。从免疫学角度看,具有代表性的克隆选择原理及独特型免疫网络原理的结合能较好地解释免疫系统中体液免疫应答的进化过程。依据这种进化思想可建立一种既能较合理反映体液免疫应答的行为特性,又能有效解决实际优化对象的免疫算法。但体液免疫应答过程的机理在免疫算法中需设计特定免疫算子加以描述,同时免疫算法的一般框架及该算法的概念需作严格定义,而且免疫算子产生的背景及免疫算法的形成过程需作具体介绍,进而给予免疫算法的基本原理。

算法的理论探讨通常包括收敛性、稳定性、鲁棒性、计算复杂度评价等。对于智能算法,大多数方法采用将算法描述为马尔可夫链(简称马氏链),并利用马氏链的一些性质进行收敛性研究。例如文献[1~10]将进化算法描述为齐次马氏链,并利用齐次马氏链的性质(如状态转移的遍历性)研究了算法的收敛性。特别文献[7]对含噪声扰动的区间适应值函数获得函数值估计结论,这种性质对探讨智能算法稳定性提供了有益参考。文献[11]将模拟退火算法描述为非齐次马氏链,并利用状态之间的转移概率,获得了该算法的收敛性结果。

智能算法的鲁棒性主要借助数值实验对参数的灵敏度进行分析,对于算法的计算复杂度分析主要依据算法在每一进化周期中所计算的次数的最高级进行度量,这种度量是依据进化群体的规模而定的。

智能算法的稳定性研究极为罕见,已有的结果所采用的方法基本上是通过构建 Lyapunov 函数或能量函数,并利用这种函数的性质获得稳定性结论,如 BP 网络及 Hopfield 网络。这种方法要求算法在搜索过程中,状态与状态之间具有确定的转移关系,或者说要求算法属于迭代式搜索算法。然而对于启发式随机搜索算法,状态之间并无确定性的转移关系, Lyapunov 稳定性方法不适合于这类算法的稳定性探讨,为此只能从随机过程理论出发,寻求探讨新的稳定性研究方法。

目前,关于基于免疫的算法研究,主要集中在工程应用方面,算法的理论探讨极为少见。对于免疫遗传算法,文献[12]利用马氏链的一些性质,获得了该算法的收敛性结果。对于利用数学方法探讨基于免疫学原理的免疫算法的理论研究尚未见报道,其主要原因在于免疫系统的内在机制极为复杂,同时既要考虑算法的结构不致复杂和具有仿生性,又要顾及算法的理论研究及实际应用效果的免疫算法框架尚未正式建立。为此以下首先建立一般性免疫算法框架,探讨其工作原理,然后

对其进行理论研究,最后对这算法的性能进行测试并与遗传算法、进化策略和免疫遗传算法进行比较,且将其用于 TSP.

体液免疫作为免疫系统的主要应答方式,细胞免疫对体液免疫起到激活和调节作用.体液免疫刻画了抗体学习、清除抗原的行为特性,这种应答方式属于一种进化过程.以下介绍体液免疫中免疫机理与免疫算法的算子模块对应关系.

体液免疫应答反映了免疫系统中抗体与抗原、抗体与抗体之间的相互作用关系,以及刻画了抗体学习抗原的行为特性.从此应答的原理分析获知,这种应答过程主要包含克隆选择、细胞克隆、记忆细胞获取、亲和突变、克隆抑制及动态平衡维持等机制,这些机制的作用关系如图 3-1 所示.此图中各免疫机制的相互作用的目的是增强免疫系统的自身防御能力,以及抗体群从识别抗原的结构模式到进化自身的模式,直到匹配抗原,以至最终中立抗原.这一过程属于抗体群的学习过程,受到这种进化方式的启发,免疫算法由此而生,其为解决优化问题而设计.图 3-1 的整个运行机制充分体现了抗体学习抗原的进化过程,构建与该图中各机理相对应的算子模块,并将这些模块按照此图中各机理的作用方式进行组合,进而获得免疫算法.在免疫算法设计中,B 细胞、克隆细胞及记忆细胞皆被视为抗体,所涉及的免疫学基本概念为:抗原、抗体、记忆细胞、自我抗体,这些概念在免疫算法中所指代的对象由表 3-1 描述.

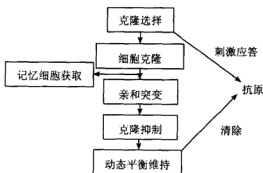


图 3-1 体液免疫机理的作用关系

表 3-1 免疫学概念与免疫算法的关系

免疫系统	免疫算法
抗原	优化问题或进化群体中最好的解
抗体	优化问题的可行解
记忆细胞	进化群体中较好的解
自我抗体	优化问题的可行解

另外免疫系统中的动态平衡维持机制涉及抗体之间的抑制与促进以及自我抗体的产生,因此为了便于免疫算法的描述,将动态平衡维持机理划分为两种机制,

即免疫选择和募集新成员。结合此节所介绍的各种机理,并依据图 3-1,免疫算法与免疫机理的关系如表 3-2 所示。

表 3-2 免疫系统机理与免疫算法的对应关系

免疫系统		免疫算法	
免疫学原理	体液免疫	免疫算子	免疫算子的含义
	克隆选择	克隆选择	进化群体中亲和力和较高的抗体被确定性选择
克隆选择原理	细胞分化繁殖	细胞克隆	被选中的抗体各繁殖一定数目的克隆
	记忆细胞获取	记忆细胞演化	分化的部分细胞作为记忆细胞更新记忆池
	亲和成熟	亲和突变	对克隆的各基因依据母体的亲和力进行突变
独特型免疫网络调节原理	克隆抑制	克隆抑制	浓度高及亲和力较低的克隆被清除
	动态平衡维持	免疫选择	依据抗体浓度及亲和力按概率随机选择抗体
		募集新成员	随机产生自我抗体插入抗体群

第二节 免疫算法的概念及工作原理

免疫算法是一种确定性和随机性选择相结合并具有勘测与开采能力的启发式随机搜索算法,它被认为是对自适应免疫应答中体液免疫的简单模拟,这种应答过程通过抗体学习抗原来完成。克隆选择使亲和力较高的抗体被确定性地选择参与进化,细胞克隆使亲和力较高的抗体依据其亲和力进行繁殖克隆细胞,其中部分克隆作为记忆细胞更新记忆池,即记忆细胞演化,其余克隆参与进化。亲和突变使克隆依据其母体的亲和力按可变概率进行变异,克隆抑制消除相同、相似及亲和力低的克隆,免疫选择使母体群参与克隆群竞争并按概率选择存活的母体或克隆,募集新成员则随机产生自我抗体插入抗体群维持群体自身平衡。这种由进化链(抗体群→克隆选择→细胞克隆及记忆细胞演化→亲和突变→免疫选择→募集新成员→新抗体群)构成的随机搜索链为进化过程,反映在数学上则为免疫算法。

免疫算法的工作原理如图 3-2,这种算法是为解决最优化问题(NBP)设计的。此图中的每一操作对应免疫系统中一种进化机制,这些操作相互作用便构成了免疫算法,其反映在免疫学上,则为体液免疫应答过程,即抗体学习抗原并最终清除抗原的过程。将此进化过程中的抗体对应优化问题(NBP)的候选解,抗原被视为问题(NBP),进而获得寻求最优解的免疫算法。

由图 3-2 可知,免疫算法作用于抗体群,而抗体群以抗体为对象进行进化。此算法包含 5 个基本要素,即抗体编码、亲和力、浓度、参数选择及免疫算子设计。

抗体编码选择为二进制编码, 目的是便于分析免疫算法的运行机制及对其进行理论探讨, 具体编码策略可参见文献 [13, 14]. 抗体的字符长度为 l , 每一个字符称为抗体的基因, 所有抗体构成抗体空间 S , 则 $|S| = 2^l$. 抗体群的规模取定为 N , 所有抗体群构成的空间称为状态空间 S^N . 由于抗体群中的抗体可相同, 因此根据可重组定理得到 $|S^N| = C_{2^l+N-1}^N$. 抗体的亲和力及浓度在下一节作具体设计. 算法的主要参数包括群体规模、克隆选择率、细胞克隆规模、克隆抑制半径、自我抗体插入群体的比率.

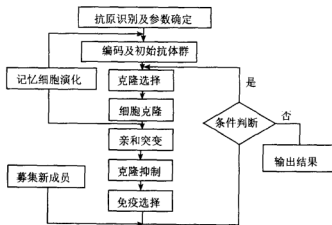


图 3-2 免疫算法的基本原理

免疫算子由该图中各操作构成, 在这里, 记忆细胞演化意指将抗体细化的部分细胞作为记忆细胞加入记忆池, 并清除相同及相似记忆细胞. 引入抗原识别及记忆细胞演化的目的在于, 对处理过的抗原或相似抗原的出现时, 提高算法寻优的快速性, 若取消此两操作, 对算法的收敛性无影响.

在这些算子中, 克隆选择将进化群体中较好的候选解确定性地选择参与进化, 提供开采更好候选解的机会. 细胞克隆及记忆细胞演化不仅为同类问题的解决提供高效解决的机会, 而且为算法的局部搜索提供必要的准备. 这一操作与亲和突变共同作用, 增强算法局部搜索能力, 使算法有更多机会探测更好的候选解. 克隆抑制促使突变的克隆群中相同或相似的克隆被确定性地清除, 其作用不仅在于保存好、中、差的克隆, 而且为免疫选择算子选择存活抗体减轻选择压力. 免疫选择的作用在于不仅给亲和力高的抗体提供更多选择机会, 而且也给亲和力及浓度皆低的抗体提供生存机会, 使得存活的抗体群具有多样性, 这一机制主要反映了抗体的促进和抑制机理及抗体选择的随机性. 募集新成员的作用在于微调群体多样性及增强全局搜索能力, 促成算法具有开放式特点, 即随时有自我抗体被引入. 这些算子相互作用, 共同合作能完成处理优化问题 (NBP) 的任务. 该算法的特点表现在如下几方面:

- (1) 抗体的选择是确定性和随机性的统一;
- (2) 细胞克隆及亲和突变的协作体现了邻域搜索及并行搜索特性;
- (3) 抗体的选择及突变受其亲和力制约, 突变概率被动态调节;
- (4) 搜索过程处于开采、探测、选择、自我调节的协调合作过程, 体现了体液免疫应答中抗体学习抗原的行为特性;
- (5) 搜索过程处于开放, 随时有自我抗体被加入进化群体, 此不仅增强群体多样性, 而且提供了产生更好解的机会;
- (6) 算法的收敛性对初始群体的分布无依赖性, 且算法对问题 (NBP) 的目标函数无连续性要求。

第三节 免疫算子及相关概念

免疫学中几个基本概念的思想在免疫算法的设计中得到有效应用, 即亲和力和相似度、浓度及激励度。在此从应用的角度, 根据算法设计的需要给出数学描述, 其中相似度及浓度的设计可由已有文献结果类似获得。

定义 3.3.1 亲和力和指抗体与抗原的匹配程度。反映在优化问题 (NBP) 上, 抗体的亲和力定义为函数 $aff: S \rightarrow (0, 1)$, $aff(x)$ 与 $f(x)$ 成反比。在此, $f(x)$ 仍表示抗体 x 对应的可行解的目标函数值。

注 亲和力和函数可选择为其他形式, 只要保证 $aff(x)$ 与 $f(x)$ 成反比。若亲和力定义为 $aff: S \rightarrow \mathbb{R}$, 则克隆突变率设计只要保证突变率在 $(0, 1)$ 上取值, 且与亲和力成反比即可。我们在这里选择

$$aff(x) = \frac{1}{1 + e^{\eta f(x)}}, \quad 0 < \eta < 1$$

定义 3.3.2 相似度指抗体与其他抗体的相似程度, 其被定义为函数 $Aff: S \times S \rightarrow [0, 1]$, 此根据信息熵理论设计。具体而言, 设 M 为含 m 个字符的字符集, 群体 G 为由 N 个长度为 l 的字符串构成的集合, 即

$$G = \{X = x_1 x_2 \cdots x_l, x_i \in M, 1 \leq i \leq l\}.$$

于是 G 中基因座 j 的信息熵定义为

$$H_j(G, N) = \sum_{i=1}^m -p_{ij} \log p_{ij}, \quad (3.3.1)$$

其中 p_{ij} 为 M 中第 i 个符号出现在基因座 j 上的概率。由信息熵理论获知, 当 $p_{ij} = \frac{1}{m}$ 时, $H_j(G, N)$ 取最大值, 即 G 中基因座 j 上出现相同信息的可能性最

小. 群体 G 的平均信息熵为

$$H(G, N) = \frac{1}{l} \sum_{j=1}^l H_j(G, N). \quad (3.3.2)$$

若 M 为二进制字符集, 则抗体 u 与抗体 v 的相似度可取为

$$\text{Aff}(u, v) = H(G, 2), G = \{u, v\}, \quad u, v \in S$$

式 (3.3.1) 刻画了群体 G 中第 j 基因座的基因信息, 式 (3.3.2) 反映了群体 G 的基因的平均信息. 当 $\text{Aff}(u, v) = 0$ 时, u, v 完全相同; 当 $\text{Aff}(u, v) = \log 2$ 时, u, v 完全匹配.

定义 3.3.3 抗体的浓度指抗体在抗体群中与其相似的抗体所占的比例, 其被定义为函数 $C: X \subset S \rightarrow [0, 1]$, 即

$$C(u) = \frac{|\{v \in X | \text{Aff}(u, v) \leq \sigma\}|}{N}, \quad (3.3.3)$$

其中 σ 为浓度阈值, $0 \leq \sigma \leq 1$, 在此称为浓度抑制半径.

由抗体浓度定义知, $0 < C(u) \leq 1$. 当 $C(u) = \frac{1}{N}$ 时, X 中不存在与 u 相似的抗体; 当 $C(u) = 1$ 时, X 中的抗体与 u 相似或相同.

定义 3.3.4 激励度是指抗体群中抗体应答抗原和被其他抗体激活的综合能力, 其可定义为函数 $\text{act}: X \subset S \rightarrow \mathbb{R}^+$,

$$\text{act}(x) = \text{aff}(x) e^{-\frac{c(x)}{\beta}}, \quad (3.3.4)$$

其中 β 为调节因子, $\beta \geq 1$. 由式 (3.3.4) 可看出, 抗体应答抗原的综合能力与其亲和力和成正比, 与其在抗体群中的浓度成反比, 此设计类似于文献 [18] 的个体聚合度设计. 从应用的角度, 此式可调节抗体群多样性.

免疫算子的设计作为免疫算法的核心内容, 将图 3-2 中各算子具体定义如下:

定义 3.3.5 克隆选择是指在给定的选择率 α 下, $0 < \alpha < 1$, 在抗体群中选择部分抗体的确定性映射, $T_s: S^N \rightarrow S$. 即设 X_0 为抗体群 X 中 $N_1 \equiv \text{round}(\alpha|X|)$ 个亲和力和较高的抗体构成的群体, 按如下概率规则选择抗体:

$$P(T_s(X)_i = X_i) = \begin{cases} 1, & X_i \in X_0, \\ 0, & X_i \notin X_0, \end{cases} \quad (3.3.5)$$

其中 $\text{round}(\cdot)$ 为取整函数, 以后出现此函数将省略其说明.

从免疫机理出发, 克隆选择仅选择群体中亲和力和较高的抗体参与繁殖及突变, 而亲和力和低的抗体仍存于免疫系统中, 并逐渐被驱除.

定义 3.3.6 设 X 为给定的抗体群, $|X| = m, m < N$, 所谓细胞克隆是指在给定的繁殖数 M 下, 抗体群 X 中的所有抗体依据各自的亲和力和繁殖率共繁殖 M 个克隆的映射. $T_c: X \subset S \rightarrow 2^S$, 它是确定性的映射, 即设 $r: S^m \rightarrow \mathbb{R}^+$ 为抗体群的繁殖率函数, $X = \{x_1, x_2, \dots, x_m\}$ 为抗体群, 则定义抗体 x_i 繁殖 m_i 个克隆, $T_c(x_i) = \text{clone}(x_i)$, 其中 $\text{clone}(x_i)$ 为 m_i 个与 x_i 相同的克隆构成的集合, m_i 由下式确定:

$$m_i = N \cdot r(X) \cdot \text{aff}(x_i), \quad \sum_{i=1}^m m_i = M \quad (3.3.6)$$

由式 (3.3.6) 知, 繁殖率函数表示为

$$r(X) = \frac{M}{N \sum_{i=1}^m \text{aff}(x_i)},$$

抗体群 X 繁殖的克隆细胞构成的群体为

$$T_c(X) = \bigcup_{i=1}^m \text{clone}(x_i)$$

定义 3.3.5 说明抗体群中较好的抗体被确定性地选择参与进化, 以便提高其自身的亲和力. 定义 3.3.6 表明, 对于给定的抗体群 X , 各抗体繁殖的克隆个数与其亲和力成正比. 细胞克隆算子是确定免疫算法运行速度的重要环节, M 过大, 则算法搜索速度较慢, M 过小, 则群体的多样性受影响.

定义 3.3.7 所谓亲和突变是指抗体空间到自身的随机映射, $T_m: S \rightarrow S$, 其作用方式是抗体按与其亲和力成反比的可变概率独立地改变自身的基因, 在此我们选择

$$P(x) = \exp(-\text{aff}(x)).$$

定义 3.3.8 所谓克隆抑制是指在抗体群中依据抗体的亲和力和相似度抑制部分抗体的确定性映射, $T_r: S^M \rightarrow S$. 克隆抑制算子的设计是, 设 X 是规模为 M 的抗体群, 依据抗体的相似度和抑制半径 σ 以及式 $\text{Aff}(u, v) < \sigma$, 将 X 划分为子群, 不妨设获 q 个子群 $P_i, 1 \leq i \leq q$, 利用处罚函数对 P_i 中亲和力低的抗体进行处罚, 进而设 X 中未被处罚的抗体构成的集合为 M_r . 于是按下列规则抑制 $M - M_r$ 个抗体

$$P(T_r(X) = u) = \begin{cases} 1, & u \in M - M_r, \\ 0, & u \in M_r. \end{cases}$$

在定义 3.3.8 中, 克隆抑制是增强群体多样性的重要环节, 其模拟抗体群中抗体之间相互抑制的机理. 由于克隆抑制通过相似度及亲和力体现, 抗体的浓度由其相似度表现, 因此克隆抑制半径与浓度抑制半径可选取相同.

定义 3.3.9 所谓免疫选择是指在抗体群中依据抗体的激励度选择抗体的随机映射, $T_{is}: S^N \rightarrow S$, 其可按概率规则

$$P\{T(X)_i = x_i\} = \frac{\text{act}(x_i)}{\sum_{x_j \in X} \text{act}(x_j)} \quad (3.3.7)$$

或

$$P\{T(X)_i = x_i\} = \frac{\exp\left(\frac{\text{act}(x_i)}{T_n}\right)}{\sum_{x_j \in X} \exp\left(\frac{\text{act}(x_j)}{T_n}\right)}, \quad T_n = \ln\left(\frac{T_0}{n} + 1\right) \quad (3.3.8)$$

选择抗体群 X 中的抗体. 式 (3.3.7) 为比例选择规则, 式 (3.3.8) 为模拟退火选择, T_0 为初始温度, n 为迭代数. 在此, 免疫选择与遗传算法中的比例选择规则及模拟退火算法中的拟退火选择规则的主要不同在于引入抗体的浓度概念, 增强群体的多样性.

注 免疫选择中的选择规则可改用其他概率选择规则, 如以抗体激励度作为概率进行选择. 从免疫机理的角度, 免疫选择反映了抗体选择的不确定性以及抗体的抑制与促进机制.

以下为了便于理论分析, 对募集新成员操作定义为算子.

定义 3.3.10 所谓募集新成员是指在抗体空间 S 中随机选择一个抗体 (即自我抗体) 的映射. 用 $T(S)$ 表示选择的抗体, 用 $T^d(S)$ 表示选择的 d 个抗体的集合.

募集新成员算子主要在于维持群体的动态平衡, 起到微调群体多样性的作用. 特别地, 当进化群体出现多样性差及全局搜索能力弱时, 自我抗体产生的随机性, 促使群体沿更好的解所在区域转移, 这种思想可由图 3-3 获知.

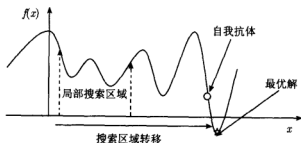


图 3-3 自我抗体对群体搜索的作用

第四节 突变规则

抗体突变是产生高亲和力抗体及多样化抗体的主要环节, 对于二进制编码的字符串, 抗体基因的突变与基本遗传算法突变规则的区别在于抗体基因的突变率

与抗体的亲和力成反比,而基本遗传算法突变率为固定不变的很小值,因此基本遗传算法中的突变规则可被采用。另外介绍几种适用于编码为多字符串的组合优化的突变算子,这些算子的抗体由字符集的元素的全排列表示,其中换位算子、逆转算子、移位算子及优质子串保留算子由文献 [15] 提出,特此注明。

(1) 换位算子: 对抗体 $A_b = (a_1, a_2, \dots, a_N)$, 以概率 p_e 随机取 r 对字符换位。当 $r = 1$ 时, 为单对字符换位; 当 $r > 1$ 时, 为多对字符换位。

(2) 逆转算子: 对抗体 $A_b = (a_1, a_2, \dots, a_N)$, 随机取两个正整数 p, q , 从 A_b 中取出一字符串 $A_1, A_1 = (a_p, a_{p+1}, \dots, a_{q-1}, a_q)$, 以 p_1 的概率对 A_1 中的各个字符首尾倒置。

(3) 移位算子: 对抗体 $A_b = (a_1, a_2, \dots, a_N)$, 随机取两个正整数 p, q , 从 A_b 中取出一字符串 $A_1, A_1 = (a_p, a_{p+1}, \dots, a_{q-1}, a_q)$, 以 p_m 的概率依次往左 (或右) 移动字符串 A_1 中和的各字符。

(4) 优质子串保留算子: 如果若干个抗体与抗原之间的亲和力都很大, 且这些抗体包含了相同的字符串 A_1 , 则在新抗体产生的过程中, 以 p_s 的概率将字符串保留下来, 操作时使 A_1 不受破坏。

以上四种突变规则的共同特点是抗体的基因突变是在抗体的基因之间进行, 突变的概率不变。对于二进制编码的字符串, 若使用以上突变操作, 当初始群体中抗体的基因几乎相同时, 所对应的算法易于陷入局部搜索, 因此在使用时应特别注意。以下给出一种突变策略。

(5) 基因块重组: 对于二进制串或由多字符集的元素构成的全排列的字符串, 同时随机选取多组长度的相等, 且各组的两块基因块互不交叉, 各组依次交换各基因位置上的基因, 如图 3-4 所示。

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
N	O	C	J	K	L	G	H	I	D	E	F	M	A	B

图 3-4 基因块互换

第五节 免疫算法描述

根据免疫算法原理图 3-2 及定义 3.3.1~3.3.10, 一般性免疫算法可描述如下:

step 1 确定参数 $N, M, \alpha, \sigma, \mu, M > \alpha N$ 。

step 2 若抗原为新抗原 (即未处理过的优化问题), 则随机产生 N 个抗体构成初始抗体群 A_0 , 记忆池 M_{set} 为空集; 若抗原为已出现过的抗原, 则从记忆池 M_{set} 中随机选择部分记忆细胞, 以及随机产生部分抗体构成规模为 N 的初始抗体群 A_0 ; 计算 A_0 中抗体的亲和力。

step 3 利用克隆选择算子在 A_n 中选择 N_1 个抗体构成群体 B_n 。

step 4 细胞克隆算子作用于 B_n 繁殖 M 个克隆, B_n 中的抗体进入记忆池 M_{set} , 并更新 M_{set} 中亲和力低的抗体.

step 5 依据亲和突变算子对各个克隆细胞进行突变, 获得突变的克隆集 C_n .

step 6 计算 C_n 中与母体不相同的克隆的亲和力, 克隆抑制算子作用于 C_n , 获得克隆集 C_n^* .

step 7 计算 $\langle A_n \oplus C_n^* \rangle$ 中抗体的激励度. 若用比例选择, 则在 $\langle A_n \oplus C_n^* \rangle$ 中选择 $N - N_3 - 1$ 个抗体, $N_3 \equiv \text{round}(\mu \cdot N)$, 其中 A_n 中亲和力最高的一个抗体 Ab 不参与选择. Ab 放入已被选中的抗体构成的群体中的最后位置, 从而获规模为 N_2 的群体 D_n , $N_2 \equiv N - N_3$. 若用模拟退火选择, 则在 $\langle A_n \oplus C_n^* \rangle$ 中选择 N_2 个抗体, 所选择的抗体构成群体 D_n . 其中 $\langle A_n \oplus C_n^* \rangle$ 为 A_n 与 C_n^* 中亲和力较高的 N 个抗体构成的集合.

step 8 由募集新成员算子任选 N_3 个自我抗体插入 D_n , 并计算此 N_3 个抗体中与 D_n 的抗体不相同的自我抗体的亲和力, 获得 A_{n+1} .

step 9 若满足终止条件, 输出结果; 否则, 返回 step3.

评注 3.5.1 Step 3~Step5 的设计思想与克隆选择算法有些相似, 主要原因在于皆从克隆选择原理出发进行设计, 但具体设计存在本质差异, 即克隆选择算法是选择亲和力最高抗体进行繁殖相同数目克隆, 未体现抗体的差异性. 而上算法中 Step 3 和 Step4 将亲和力较高抗体繁殖数目不等的克隆子群, 且克隆群的总规模受到限制, 这两步的结合较合理地体现了克隆选择原理的机理, 同时有助于增强算法群体的多样性, 防止早熟现象.

评注 3.5.2 Step 6~Step 8 模拟免疫系统的动态平衡机制, Step 7 应用了抗体选择的不确定性特征及抗体促进与抑制机理. 这种机理在免疫遗传算法的选择算子中已被应用, 以上算法描述的免疫选择与免疫遗传算法的选择算子的主要区别在于免疫选择的设计更具有灵活性. 抗体激励度刻画了亲和力与浓度的关系, 亲和力反映了抗原与抗体的关系, 抗原可指代优化问题本身, 也可指代进化群体中最好的抗体. 同时抗体激励度可作为其选择概率, 使此抗体按这种概率进行选择.

算法中的参数分析作为鲁棒性分析将在第十一节介绍. 这些参数的范围要求是, $M > N_1, 0 < \alpha, \sigma < 1, 0.05 \leq \mu \leq 0.08$. 参数 μ 的选择是根据免疫系统维持自我平衡所需新抗体数而设定的^[16], 其反映在算法上, 若 μ 过大, 则出现波动现象; 若 μ 趋于 0, 则导致群体多样性不足. 因此应合理选择 μ 的取值范围, 这种观点在后面的鲁棒性分析中得到证实. 借助于算子描述, 该算法可描述为随机搜索序列 $\{A_n, n \geq 0\}$,

$$A_n = T_{is}(T_{rs}(A_{n-1}) \oplus A_{n-1}) \cup T^{N_3}(S), \quad (3.5.1)$$

其中 T_{rs} 表示算子 T_r, T_m, T_c, T_s 的复合算子, 即

$$T_{rs} = T_r \circ T_m \circ T_c \circ T_s, \quad (3.5.2)$$

简记为

$$z[n] = \min(f(A_n)), \quad f(A_n) = \{f(x) : x \in A_n\}. \quad (3.5.3)$$

在此, $z[n]$ 表示抗体群 A_n 中亲和力最高的抗体所对应的候选解的目标值, 于是 $\{z[n], n \geq 0\}$ 构成随机序列. 依据算子 T_{is} 的具体设计, 一般性免疫算法可包括两种不同类型的算法, 即当免疫算子 T_{is} 选为比例选择时, 以上算法可描述为齐次马尔可夫链, 马尔可夫链简记为马氏链. 为便于表述, 称此时的免疫算法为齐次免疫算法 (HIA: homogeneous immune algorithm). 当免疫算子 T_{is} 选为模拟退火算子时, 由于此算子与时间有关, 因此称此时的免疫算法为非齐次免疫算法 (IHIA: inhomogeneous immune algorithm). 这两种算法在实际应用效果及理论研究方面皆存在较大差异, 因此有必要对这两种算法同时研究, 也可从一侧面反映出研究可描述为齐次马氏链及非齐次马氏链的算法的一些思想方法. 从 step 8 可看出, A_{n+1} 由两部分构成. 一部分为 A_n 经算子 T_{rs}, T_{is} 作用, 所获的 $N - N_3$ 个抗体构成的群体, 记为 A_{n+1}^1 ; 另一部分为随机产生的 N_3 个新抗体, 记为 A_{n+1}^2 , 则有

$$A_{n+1}^1 = T_{is}(T_{rs}(A_n) \oplus A_n), \quad A_{n+1}^2 = T^{N_3}(S).$$

对于非齐次免疫算法还可考虑克隆细胞的突变概率与时间有关. 引入如下说法: S^N 中每一元素称为状态, 每一状态由 N 个抗体组成. 用 M^* 表示由 $f(x)$ 在 S 上取最小值的所有抗体构成, 其最小值记为 f^* . 由于 M^* 仅包含有限个不同的最优解, 所以不妨设 M^* 中抗体之间的相似度大于 σ . 用 $P_i(X, Y)$ 表示状态 X 经算子 T_i 转移为状态 Y 的概率, $i = s, c, m, r$. 为便于后续叙述, 引入下列记号:

$$S_\sigma^N = \{Y = (Y_1, Y_2, \dots, Y_N) \in S^N : \text{Aff}(Y_i, Y_j) > \sigma, 1 \leq i, j \leq N, i \neq j\},$$

$$S_{\sigma^2, N_3}^{N_2, N_3} = \{Z = (Z_1, Z_2) \in S^N : Z_1 \in S_{\sigma^2}^{N_2}, Z_2 \in S^{N_3}\},$$

$$a_n \equiv P(A_n \cap M^* = \phi), \quad \text{aff}(I) = \max(\text{aff}(x), x \in I), I \in S^N.$$

第六节 算法收敛性概念

以下给出随机序列收敛性的定义, 其中定义 3.6.1~3.6.4 选自文献 [1], 定义 3.6.5 选自文献 [5].

定义 3.6.1 若序列 $\{A_n, n \geq 0\}$ 满足

$$\lim_{n \rightarrow \infty} P(A_n \cap M^* \neq \phi) = 1,$$

则称此序列概率弱收敛.

定义 3.6.2 若随机序列 $\{A_n, n \geq 0\}$ 满足

$$P(\lim_{n \rightarrow \infty} [A_n \cap M^* \neq \phi]) = 1,$$

则称此序列几乎处处弱收敛.

定义 3.6.3 若随机序列 $\{A_n, n \geq 0\}$ 满足

$$\lim_{n \rightarrow \infty} P(A_n \subset M^*) = 1,$$

则称此序列概率强收敛.

定义 3.6.4 若随机序列 $\{A_n, n \geq 0\}$ 满足

$$P\left(\lim_{n \rightarrow \infty} [A_n \subset M^*]\right) = 1,$$

则称此序列几乎处处强收敛.

定义 3.6.5 若序列 $\{z[n], n \geq 0\}$ 的期望满足

$$\lim_{n \rightarrow \infty} E[|z[n] - f^*|] = 0,$$

则称随机序列 $\{A_n, n \geq 0\}$ 按期望收敛.

定义 3.6.6 对于任意给定的 $\delta > 0$, 如果对 $\forall \varepsilon > 0, \exists N(\varepsilon) > 0$, 当 $n > N(\varepsilon)$ 时, 序列 $\{z[n], n \geq 0\}$ 满足

$$P(|z[n] - f^*| \leq \delta) \geq 1 - \varepsilon,$$

则称随机序列 $\{A_n, n \geq 0\}$ 依概率收敛.

由定理 3.1.1^[1] 获知, 随机序列 $\{A_n, n \geq 0\}$ 具有下列收敛关系:

(1) 几乎处处强收敛 \Rightarrow 几乎处处弱收敛 \Rightarrow 概率弱收敛;

(2) 几乎处处强收敛 \Rightarrow 概率强收敛 \Rightarrow 概率弱收敛.

定理 3.6.1 若随机序列 $\{A_n, n \geq 0\}$ 满足

$$a_{n+1} \equiv P(A_{n+1} \cap M^* \neq \phi | A_n \cap M^* = \phi) \geq \delta, 0 < \delta < 1,$$

$$b_{n+1} \equiv P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) \leq \beta_n,$$

其中

$$\sum_{n=0}^{\infty} \beta_n < \infty,$$

则序列 $\{A_n, n \geq 0\}$ 是概率弱收敛.

证明 由假设及全概率公式获知

$$\begin{aligned} P(A_{n+1} \cap M^* = \phi) &= (1 - a_n)P(A_n \cap M^* = \phi) + b_n P(A_n \cap M^* \neq \phi) \\ &\leq (1 - \delta)P(A_n \cap M^* = \phi) + \beta_n. \end{aligned}$$

进而通过归纳得到

$$P(A_{n+1} \cap M^* = \phi) \leq (1 - \delta)^{n+1} a_0 + \sum_{k=0}^n \beta_{n-k} (1 - \delta)^k \leq b(1 - \delta)^n,$$

其中

$$\hat{a}_0 = (1 - \frac{|M^*|}{|S|})^N, \quad b \equiv (1 - \delta)\hat{a}_0 + \sum_{k=0}^{\infty} \frac{\beta_k}{(1 - \delta)^k}.$$

由假设有 $b < +\infty$, 从而

$$\lim_{n \rightarrow \infty} P(A_n \cap M^* = \phi) = 0,$$

故由定义 3.6.1 知该定理结论成立. 证毕.

另外, 按期望收敛与依概率收敛具有下面的结论.

定理 3.6.2 若随机序列 $\{A_n, n \geq 0\}$ 按期望收敛, 则必依概率收敛.

证明 对任意随机变量 ς , 若期望 $|E[\varsigma]| < \infty$, 则必有

$$P(\varsigma > \delta) \leq \frac{1}{\delta} \int_{[\varsigma > \delta]} \varsigma \, dP \leq \frac{1}{\delta} E[\varsigma],$$

故有

$$P(|z[n] - f^*| > \delta) \leq \frac{1}{\delta} E[|z[n] - f^*|],$$

从而, 当 $\{A_n, n \geq 0\}$ 按期望收敛时, 对 $\forall \varepsilon > 0$, 必 $\exists N(\varepsilon) > 0$, 使得当 $n > N(\varepsilon)$ 时,

$$E[|z[n] - f^*|] < \varepsilon.$$

进而

$$\begin{aligned} P(|z[n] - f^*| \leq \delta) &= 1 - P(|z[n] - f^*| > \delta) \\ &\geq 1 - \frac{1}{\delta} E[|z[n] - f^*|] > 1 - \frac{\varepsilon}{\delta}, \end{aligned}$$

故结论成立. 证毕.

第七节 免疫算子性质及齐次免疫算法收敛性

免疫算法中克隆选择、细胞克隆、亲和突变及克隆抑制算子作为免疫算法的基本算子, 这些算子的复合算子 T_{rs} 、 $T_{ms} = T_m \circ T_c \circ T_s$ 及 T_m 具有如下性质.

定理 3.7.1 对 $\forall X, Y \in S^M$, 亲和突变算子 T_m 必满足

$$P_m(X, Y) > 0.$$

证明 不妨设 $X = (X_1, X_2, \dots, X_M)$, $Y = (Y_1, Y_2, \dots, Y_M)$. 由亲和力及亲和突变算子的定义获知, 存在 $\alpha^* > 0$, 使得 $\forall X_i$ 的突变概率

$$p(X_i) \equiv P(\text{aff}(X_i)) \geq P(\alpha^*).$$

由于 X_i 及 Y_i 皆是长度为 l 的二进制字符串, 因此 X_i 变异为 Y_i 的概率为

$$m(i) \equiv p(X_i)^{d(X_i, Y_i)} (1 - p(X_i))^{l-d(X_i, Y_i)}, i = 1, 2, \dots, M,$$

其中 $d(X_i, Y_i)$ 表示 X_i 与 Y_i 的海明距离, 从而存在常数 $\delta^*, 0 < \delta^* < 1$, 使得

$$P_m(X, Y) = \prod_{i=1}^M m(i) \geq \delta^*,$$

故该定理的结论成立. 证毕.

推论 3.7.1 对于 $\forall X \in S^N, Y \in S^M$, 复合算子 T_{ms} 具有下列性质:

$$P(T_{ms}(X) = Y) > 0.$$

证明 由于克隆选择算子为确定性算子, 因此对于 $\forall X \in S^N$, 必存在唯一 $X_0 \in S^{N_1}$, 使得

$$P(T_s(X)_i = X_{0i}) = 1, \quad X_0 = (X_{01}, X_{02}, \dots, X_{0N_1}),$$

从而对于 $Z \in S^M, Z \cup X_0 = X_0$, 有

$$P(T_c \circ T_s(X) = Z) = 1.$$

于是, 由定理 3.7.1 及马氏链的 K-C 方程导出

$$\begin{aligned} P(T_{ms}(X) = Y) &= \sum_{J \in S^M} P(T_c \circ T_s(X) = J) \cdot P(T_m(J) = Y) \\ &= P(T_m(Z) = Y) \geq P(\alpha^*). \end{aligned}$$

证毕.

定理 3.7.2 对于 $\forall X \in S^N$, 下式成立:

$$P(T_{rs}(X) = Y) \begin{cases} > 0, & Y \in S_{\sigma}^{\leq M}, \\ 0, & Y \in S^{\leq M} \setminus S_{\sigma}^{\leq M}, \end{cases}$$

其中 $S^{\leq M}$ 表示群体规模不超过 M 的抗体群构成的集合.

证明 由 K-C 方程有

$$P(T_{rs}(X) = Y) = \sum_{I \in S^M} P(T_{ms}(X) = I) \cdot P_r(I, Y). \quad (3.7.1)$$

由克隆抑制算子的定义知, 克隆群经过抑制后, 群体中相同及相似抗体被消除, 剩下的克隆之间的相似度皆大于 σ . 因此对于 $\forall K \in S^M$, 若 $Y \in S^{\leq M} \setminus S_{\sigma}^{\leq M}$, 必有

$P_r(K, Y) = 0$, 进而由推论 3.7.1 及式 (3.7.1) 推出, 对于 $\forall X \in S^N, Y \in S^{\leq M} \setminus S_{\sigma}^{\leq M}$, 有

$$P(T_{sr}(X) = Y) = 0.$$

另一方面, 由于算子 T_r 为确定性映射, 因此由推论 3.7.1 推出, 当 $X \in S^N, Y \in S_{\sigma}^{\leq M}$ 时,

$$P(T_{sr}(X) = Y) \geq \sum_{Y \cup Z = Y, Z \in S^M} P(T_{sm}(X) = Z) P_r(Z, Y) > 0.$$

证毕.

现将齐次免疫算法的收敛性定理描述如下.

定理 3.7.3 对于任意初始分布, 当克隆选择率 $0.1 < \alpha < 1$, 抑制半径 $0 \leq \sigma \leq 0.5$ 时, 算法 HIA 是概率弱收敛.

证明 参数 α, σ 的取值范围由齐次免疫算法的鲁棒性分析获知. 由免疫算法的 step8 可知, 若第 n 时刻 A_n 为抗体群 $X = \{X_i, 1 \leq i \leq N\}$, 不妨设此抗体群中亲和力最高的一个抗体为 X_N , $n+1$ 时刻 A_{n+1} 为抗体群 $Y = \{Y_i, 1 \leq i \leq N\}$, 则

$$A_{n+1}^1 = (Y_1, Y_2, \dots, Y_{N_2-1}) \equiv \hat{Y}_1, \quad Y_{N_2} = X_N,$$

$$A_{n+1}^2 = T^{N_3}(S) = (Y_{N_2+1}, \dots, Y_N) \equiv \hat{Y}_2.$$

于是有

$$P(A_{n+1} = Y | A_n = X) = \begin{cases} P(T_{is}(\langle T_{rs}(X) \oplus X \rangle) = \hat{Y}_1) \cdot P(T^{N_3}(S) = \hat{Y}_2), & Y_{N_2} = X_N, \\ 0, & Y_{N_2} \neq X_N. \end{cases} \quad (3.7.2)$$

以下分两种情形考虑状态转移概率:

(1) 当 $X \cap M^* \neq \phi, Y \cap M^* = \phi$ 时, 由式 (3.7.2) 知

$$P(A_{n+1} = Y | A_n = X) = 0,$$

从而

$$\begin{aligned} & P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) \\ &= \sum_{X \cap M^* = \phi} \sum_{Y \cap M^* = \phi} P(A_{n+1} = Y | A_n = X) = 0. \end{aligned} \quad (3.7.3)$$

(2) 当 $X \cap M^* = \phi, Y \subset M^*$ 时,

$$P(T^{N_3}(S) = \hat{Y}_2) = \left(\frac{|M^*|}{|S|} \right)^{N_3}, \quad (3.7.4)$$

$$P(T_{is}(\langle T_{rs}(X) \oplus X \rangle) = \hat{Y}_1) = \sum_{Z \in S_{\sigma}^{\leq M}, \hat{Y}_1 \subset \langle Z \oplus X \rangle} P(T_{is}(\langle Z \oplus X \rangle) = \hat{Y}_1). \quad (3.7.5)$$

对于 $\hat{Y}_1 \subset \langle Z \oplus X \rangle$,

$$P(T_{is}(\langle Z \oplus X \rangle) = \hat{Y}_1) = \prod_{y \in \hat{Y}_1} \frac{\text{aff}(y) \cdot e^{-\frac{c(y)}{\beta}}}{\sum_{w \in \langle Z \oplus X \rangle} \text{aff}(w) \cdot e^{-\frac{c(y)}{\beta}}} \geq \left(\frac{1}{N} \cdot e^{-\frac{1}{\beta}}\right)^{N_2-1}. \quad (3.7.6)$$

由定理 3.7.1、定理 3.7.2、推论 3.7.1、式 (3.7.2) 及式 (3.7.4)~(3.7.6),

$$P(A_{n+1} = Y | A_n = X) \geq \delta^* \left(\frac{|M^*|}{|S|} \right)^{N_3} (Ne^{\frac{1}{\beta}})^{1-N_2} \equiv \delta, 0 < \delta < 1, \quad (3.7.7)$$

从而由式 (3.7.7) 得

$$P(A_{n+1} \cap M^* \neq \phi | A_n \cap M^* = \phi) \geq P(A_{n+1} = Y | A_n = X) \geq \delta. \quad (3.7.8)$$

进而由式 (3.7.3)、式 (3.7.8) 及定理 3.6.1 得到该定理的结论. 证毕.

评注 3.7.1 定理 3.7.3 是基于免疫算子的性质, 并利用全概率公式及 K-C 方程获证. 由此定理的证明可以看出, 抗体的浓度对算法的收敛性无影响, 但影响算法的搜索性, 抗体的浓度仅起到调节群体多样性的作用.

以下考虑序列 $\{z[n] - f^*, n \geq 0\}$ 按期望的收敛性, 不妨记 $\zeta_n \equiv |z[n] - f^*|, n \geq 0$.

定理 3.7.4 对于任意初始分布, 当克隆选择率 $0.1 < \alpha < 1$, 抑制半径 $0 \leq \sigma \leq 0.5$ 时, 算法 HIA 是按期望收敛, 并成立下式

$$E[\zeta_n] \leq (f_{\max} - f^*) \left(1 - \frac{|M^*|}{|S|} \right)^N (1 - \delta)^n, \quad (3.7.9)$$

其中

$$f_{\max} = \max\{f(x), x \in S\}.$$

证明 参数的选择由鲁棒性分析获知. 由定理 3.7.3 的式 (3.7.3)、(3.7.8) 及全概率公式知

$$a_n \leq \left(1 - \frac{|M^*|}{|S|} \right)^N (1 - \delta)^n. \quad (3.7.10)$$

由于该算法含有最优保留策略, 因此

$$E[\zeta_{n+1} | A_n \cap M^* \neq \phi] = 0. \quad (3.7.11)$$

另外易得

$$E[\zeta_{n+1}] = E[z[n+1]] - f^* \leq f_{\max} - f^*, \quad (3.7.12)$$

于是结合式 (3.7.10)~(3.7.12) 及下式

$$\begin{aligned} E[\zeta_{n+1}] &= E[\zeta_{n+1}|A_n \cap M^* = \phi]P(A_n \cap M^* = \phi) \\ &\quad + E[\zeta_{n+1}|A_n \cap M^* \neq \phi]P(A_n \cap M^* \neq \phi), \end{aligned}$$

并通过归纳可得式 (3.7.9) 成立, 进而由 $0 < \delta < 1$ 得该算法是按期望收敛. 证毕.

第八节 非齐次免疫算法收敛性

鉴于算法 IHIA 中免疫选择算子与时间 n 有关, 探讨其收敛性与齐次情形有所不同, 但皆含相同的几种基本算子, 因此定理 3.7.1、3.7.2 及推论 3.7.1 对于非齐次情形仍然成立.

定理 3.8.1 对于任意初始分布, 当 $\alpha \geq 0.1$, $0 \leq \sigma \leq 0.5$, $\alpha \neq 0.2$ 时, 算法 IHIA 是概率弱收敛.

证明 参数 α, σ 的取值范围由算法 IHIA 的鲁棒性分析导出. 由该算法的描述获知, 若 $n+1$ 时刻 A_{n+1} 为抗体群 $Y = (\hat{Y}_1, \hat{Y}_2)$, $\hat{Y}_1 \in S^{N_2}$, $\hat{Y}_2 \in S^{N_3}$, 则 $A_{n+1}^1 = \hat{Y}_1$, $A_{n+1}^2 = \hat{Y}_2$. 为便于叙述, 置

$$P(Z, X, \hat{Y}_1, n) \equiv P(T_{is}(\langle Z \oplus X \rangle) = \hat{Y}_1 | T_{rs}(X) = Z, A_n = X). \quad (3.8.1)$$

(1) 当 $X \cap M^* \neq \phi$, $Y \cap M^* = \phi$ 时,

$$P(T^{N_3}(S) = \hat{Y}_2) = \left(1 - \frac{|M^*|}{|S|}\right)^{N_3}. \quad (3.8.2)$$

由 K-C 方程及式 (3.8.1)~(3.8.2) 有

$$P(A_{n+1} = Y | A_n = X) = \sum_{Z \in S^{N_2}, \hat{Y}_1 \subset \langle Z \oplus X \rangle} P(Z, X, \hat{Y}_1, n) \cdot \left(1 - \frac{|M^*|}{|S|}\right)^{N_3}. \quad (3.8.3)$$

由于抗体的亲和力与其目标函数值成反比, 因此

$$\text{aff}(x^*) = \max(\text{aff}(x), x \in S), \forall x^* \in M^*.$$

又 β 为调节因子, 因此当 $\beta > \frac{1}{\beta_0}$, $\beta_0 = \min_{y \in S \setminus M^*} \ln \frac{\text{aff}(x^*)}{\text{aff}(y)}$, $x^* \in M^* \cap \langle Z \oplus X \rangle$ 时, 对于 $\forall y_0 \in \hat{Y}_1 \subset \langle Z \oplus X \rangle$, 必有

$$\text{aff}(y_0) < \text{aff}(x^*) \exp\left(-\frac{1}{\beta}\right).$$

于是

$$\begin{aligned}
 P(Z, X, \hat{Y}_1, n) &= \prod_{y \in \hat{Y}_1} \frac{\exp\left(\frac{\text{act}(y)}{T_n}\right)}{\sum_{w \in \langle Z \oplus X \rangle} \exp\left(\frac{\text{act}(w)}{T_n}\right)} \\
 &\leq \frac{\exp\left(\frac{\text{aff}(y_0)}{T_n} \cdot \exp\left(-\frac{c(y_0)}{\beta}\right)\right)}{\sum_{w \in \langle Z \oplus X \rangle} \exp\left(\frac{\text{aff}(w)}{T_n} \cdot \exp\left(-\frac{c(w)}{\beta}\right)\right)} \\
 &\leq \exp\left(-\frac{1}{T_n} \left(\text{aff}(x^*) \cdot \exp\left(-\frac{c(x^*)}{\beta}\right) - \text{aff}(y_0) \cdot \exp\left(-\frac{c(y_0)}{\beta}\right)\right)\right) \\
 &\leq \exp\left(-\frac{1}{T_n} \left(\text{aff}(x^*) \exp\left(-\frac{1}{\beta}\right) - \text{aff}(y_0)\right)\right) \leq \exp\left(-\frac{\rho_0}{T_n}\right),
 \end{aligned} \tag{3.8.4}$$

其中

$$\rho_0 = \min \left\{ \text{aff}(x^*) \exp\left(-\frac{1}{\beta}\right) - \text{aff}(y) : y \in S \setminus M^* \right\}.$$

结合式 (3.8.2)~(3.8.4) 得到

$$P(A_{n+1} = Y | A_n = X) \leq |S^{\leq M}| \left(1 - \frac{|M^*|}{|S|}\right)^{N_3} \exp\left(-\frac{\rho_0}{T_n}\right) \equiv \hat{\delta}_n \rightarrow 0, n \rightarrow \infty. \tag{3.8.5}$$

于是

$$\begin{aligned}
 &P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) \\
 &= \sum_{X \in S^N, X \cap M^* \neq \phi} \sum_{Y \in S^N, Y \cap M^* = \phi} P(A_{n+1} = Y | A_n = X) \leq |S^N|^2 \hat{\delta}_n \equiv \beta_n.
 \end{aligned} \tag{3.8.6}$$

(2) 当 $X \cap M^* = \phi, Y \subset M^*$ 时, 若 $Z \in S_{\sigma}^{\leq M}, T_{sr}(X) = Z, \hat{Y}_1 \subset \langle Z \oplus X \rangle$, 则 $\hat{Y}_1 \subset Z$.

①若 $\sigma = 0$, 则克隆抑制算子对算法的收敛性无贡献, 此时 $c(x) = 0, \forall x \in \langle Z \oplus X \rangle$, 从而

$$P(Z, X, \hat{Y}_1, n) = \prod_{y \in \hat{Y}_1} \frac{\exp\left(\frac{\text{act}(y)}{T_n}\right)}{\sum_{w \in \langle Z \oplus X \rangle} \exp\left(\frac{\text{act}(w)}{T_n}\right)} \geq \frac{1}{N^{N_2}} \equiv \delta_1. \tag{3.8.7}$$

②若 $\sigma > 0$, 由克隆抑制算子的定义知, 对于 $\forall y^* \in \hat{Y}_1 \subset Z \cap M^*, y^*$ 在 $\langle Z \oplus X \rangle$ 中的浓度必为 0, 即 $c(y^*) = 0$. 不妨设 $Z \neq \langle Z \oplus X \rangle$, 则必有 $y_0 \in \langle Z \oplus X \rangle$ 使得

$0 < c(y_0) < 1$, 从而

$$P(Z, X, \hat{Y}_1, n) > \left(\frac{\exp\left(\frac{\text{aff}(y^*)}{T_n}\right)}{N \exp\left(\frac{\text{aff}(y^*)}{T_n}\right)} \right)^{N_2} = \delta_1. \quad (3.8.8)$$

另外,

$$P(T^{N_3}(S) = \hat{Y}_2) = \left(\frac{|M^*|}{|S|} \right)^{N_3}. \quad (3.8.9)$$

于是结合定理 3.7.1、定理 3.7.2、推论 3.7.1、式 (3.8.7)~(3.8.9) 有

$$\begin{aligned} P(A_{n+1} \cap M^* \neq \phi | A_n \cap M^* = \phi) &\geq P(A_{n+1} = Y | A_n = X) \\ &\geq \delta^* \delta_1 \left(\frac{|M^*|}{|S|} \right)^{N_3} \equiv \hat{\delta}. \end{aligned} \quad (3.8.10)$$

又由于

$$\begin{aligned} \lim_{n \rightarrow \infty} \sqrt[n]{\beta_n} &= \lim_{n \rightarrow \infty} \sqrt[n]{\hat{\delta}_n} \\ &= \lim_{n \rightarrow \infty} \sqrt[n]{\exp\left(-\frac{\rho_0}{T_n}\right)} = \exp\left(-\frac{\rho_0}{T_0}\right) < 1, \end{aligned}$$

则由正项级数的柯西判别准则得到

$$\sum_{n=0}^{\infty} \beta_n < \infty.$$

进而由 (3.8.6)、式 (3.8.10) 及定理 3.6.1 知该定理的结论成立. 证毕.

以下考虑算法 IHIA 按期望的收敛性

定理 3.8.2 对于任意初始分布, 当 $\alpha \geq 0.1$, $0 \leq \sigma \leq 0.5$, $\alpha \neq 0.2$ 时, 算法 IHIA 按期望收敛, 并成立下式:

$$E[\zeta_n] \leq b(f_{\max} - f^*)(1 - \hat{\delta})^n, \quad (3.8.11)$$

其中

$$b = (1 - \hat{\delta}) \left(1 - \frac{|M^*|}{|S|} \right)^N + \sum_{k=0}^{\infty} \frac{\beta_k}{(1 - \hat{\delta})^k}.$$

证明 由定理 3.8.1 的证明及定理 3.6.1 有

$$P(A_{n+1} \cap M^* = \phi) \leq b(1 - \hat{\delta})^n. \quad (3.8.12)$$

若 $A_{n+1} \cap M^* \neq \phi$, 则 $z[n+1] = f^*$, 从而

$$E[\zeta_{n+1} | A_{n+1} \cap M^* \neq \phi] = 0. \quad (3.8.13)$$

于是, 由式 (3.8.12)、(3.8.13) 及

$$E[\zeta_{n+1}] \leq f_{\max} - f^*$$

和

$$\begin{aligned} E[\zeta_{n+1}] &= E[\zeta_{n+1} | A_{n+1} \cap M^* = \phi] P(A_{n+1} \cap M^* = \phi) \\ &\quad + E[\zeta_{n+1} | A_{n+1} \cap M^* \neq \phi] P(A_{n+1} \cap M^* \neq \phi), \end{aligned} \quad (3.8.14)$$

推出式 (3.8.11) 成立, 进而该结论成立. 证毕.

第九节 免疫算法收敛速度分析

第七、八节研究了免疫算法的收敛性, 探讨思路均建立在以抗体群为状态的基础上进行研究. 以下以抗体群集合作为状态对免疫算法的收敛速度进行估计.

引理 3.9.1^[11] 若

$$P = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$$

是随机矩阵, 其中 C 为 m 维严格随机方阵 (即 $C = (c_{ij}), c_{ij} > 0$), R 为 $(n-m) \times m$ 阶矩阵, T 为 $(n-m) \times (n-m)$ 阶方阵, $R \neq 0, T \neq 0$, 则

$$P^\infty = \begin{bmatrix} C^\infty & 0 \\ R^\infty & 0 \end{bmatrix}$$

是一个稳定的随机阵 (即 P^∞ 是随机的, 且所有的行对应的元素相同), 其中

$$R^\infty = \lim_{n \rightarrow \infty} \sum_{i=0}^{k-1} T^i R C^{k-i}, P^\infty = [1, 1, \dots, 1]^T [p_1, p_2, \dots, p_n],$$

$$\sum_{i=1}^n p_{ij} = 1, p_j = \lim_{k \rightarrow \infty} p_{ij}^{(k)} \begin{cases} > 0, & 1 \leq j \leq m, \\ = 0, & m+1 \leq j \leq n. \end{cases}$$

推论 3.9.1 在引理 3.9.1 的假设下, 若 $m=1, C=1, R>0$, 则

$$P^\infty = \begin{bmatrix} 1 & 0 \\ e & 0 \end{bmatrix},$$

其中 $e = [1, 1, \dots, 1]^T$.

证明 由于 P 为随机矩阵以及 $R>0$, 故 $\|T\|_\infty < 1$, 从而

$$\sum_{i=0}^{\infty} T^i = (1 - T)^{-1},$$

$\|T\|_\infty$ 表示矩阵 T 的 $\|\cdot\|_\infty$ 范数. 于是由引理 3.9.1 有

$$P^\infty = \begin{bmatrix} 1 & 0 \\ (1-T)^{-1}R & 0 \end{bmatrix}.$$

令

$$(1-T)F = R,$$

则由矩阵 $I-T$ 的可逆性知此方程有唯一解. 又由 P 为随机矩阵, 则 $F = e$ 为此方程的解, 进而结论成立. 证毕.

设 $G \equiv \{g \in \mathbb{R}^+ \mid g = \text{aff}(x), x \in S\}$, G 中互不相同的个数为 \hat{M} , 不妨设 $G = \{g_i > g_j : 1 \leq i < j \leq \hat{M}\}$. 显然由 $|S| = 2^l$, 可知 $\hat{M} \leq 2^l$. 记 $Z_i = \{I \in S^N : \text{aff}(I) = g_i\}$, $z_i \equiv |Z_i|$, 则有

$$S^N = \bigcup_{i=1}^{\hat{M}} Z_i, \quad Z_i \cap Z_j = \emptyset, i \neq j.$$

用 Z_{ik} 表示 Z_i 中第 k 个元素, $P(Z_{ik}, Z_{il})$ 表示 Z_{ik} 转移为 Z_{il} 的状态转移概率, $P(Z_{ik}, Z_j)$ 表示 Z_{ik} 转移为 Z_j 中元素的种群转移概率, $P(Z_i, Z_j)$ 表示在以种群集合为状态下, Z_i 转变为 Z_j 的状态转移概率.

定理 3.9.1 对于算法 HIA, 必成立

$$P(A_{n+1} \in Z_k | A_n \in Z_i) \begin{cases} > 0, & k \leq i, \\ = 0, & k > i. \end{cases} \quad (3.9.1)$$

证明 (1) 若 $k > i$, 即 $\text{aff}(Z_k) < \text{aff}(Z_i)$ 时, 由于齐次免疫算法含有最优保存策略, 则对 $\forall Z_{ij} \in P_i, Z_{kl} \in P_k$, 有

$$P(Z_{ij}, Z_{kl}) = P(A_{n+1} = Z_{kl} | A_n = Z_{ij}) = 0. \quad (3.9.2)$$

又因为

$$P(A_{n+1} \in Z_k | A_n \in Z_i) = \sum_{j=1}^{z_i} P(Z_{ij}, Z_k) = \sum_{j=1}^{z_i} \sum_{l=1}^{z_k} P(Z_{ij}, Z_{kl}), \quad (3.9.3)$$

故由式 (3.9.2)、(3.9.3) 知, 当 $k > i$ 时,

$$P(A_{n+1} \in Z_k | A_n \in Z_i) = 0. \quad (3.9.4)$$

(2) 若 $k \leq i$, 即 $\text{aff}(Z_k) \geq \text{aff}(Z_i)$ 时, 由 Z_k 的定义, 必存在正整数 l_0 , 使得 $l_0 \leq z_k$, $Z_{kl_0} \equiv (Z_{kl_01}, Z_{kl_02}) \in Z_k, Z_{kl_01} \in S_{\sigma}^{N_2}, Z_{kl_02} \in S^{N_3}$. 进而对 $Z \in S^{\leq M}, Z_{ij} \in$

Z_i , 当 $Z_{kl_01} \subset \langle Z \oplus Z_{ij} \rangle$ 时, 下式成立:

$$P(T_{is} \langle Z \oplus Z_{ij} \rangle = Z_{kl_01}) = \prod_{y \in Z_{kl_01}} \frac{\text{aff}(y) \cdot \exp\left(-\frac{c(y)}{\beta}\right)}{\sum_{w \in \langle Z \oplus X \rangle} \text{aff}(w) \cdot \exp\left(-\frac{c(w)}{\beta}\right)} > 0, \quad (3.9.5)$$

从而由定理 3.7.2、式 (3.9.3)、式 (3.9.5) 及下式

$$P(T^{N_3}(S) = Z_{kl_02}) = \frac{1}{|S|^{N_3}}$$

可推出当 $k \leq i$ 时,

$$P(A_{n+1} \in Z_k | A_n \in Z_i) > 0. \quad (3.9.6)$$

故由式 (3.9.4) 及式 (3.9.6) 知式 (3.9.1) 成立. 证毕.

由于算法 HIA 为齐次马尔可夫链 $\{A_n, n \geq 0\}$, 因此引入记号 $p_{ij} \equiv P(A_{n+1} \in Z_k | A_n \in Z_i)$, 从而借助于定理 3.9.1 可得以抗体群集合为状态的状态转移矩阵

$$P = \begin{bmatrix} p_{11} & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ p_{\hat{M}1} & p_{\hat{M}2} & \cdots & p_{\hat{M}\hat{M}} \end{bmatrix}, \quad p_{ij} > 0, j \leq i, \sum_{j=1}^i p_{ij} = 1, \quad 1 \leq i \leq \hat{M}.$$

记

$$a_{ni} = P(A_n \in Z_i), i = 1, 2, \cdots, \hat{M}, z_2(0) = (a_{02}, a_{03}, \cdots, a_{0\hat{M}}), \\ \hat{z}(n) = (a_{n1}, a_{n2}, \cdots, a_{n\hat{M}}),$$

于是结合定理 3.9.1 及推论 3.9.1 易证下列定理成立

定理 3.9.2 算法 HIA 的概率分布序列 $\{\hat{z}(n), n \geq 0\}$ 收敛于 $z(\infty) = z(0)P^\infty$ 的速度估计为

$$\|z[n] - z(\infty)\| \leq M_0 \|T\|^n,$$

其中

$$M_0 = 1 + \|R\| \cdot \|(1-T)^{-1}\| \cdot \|z_2(0)\|.$$

值得一提的是, 定理 3.9.2 提供了探讨算法 HIA 收敛性的另一种方法, 定理 3.7.3 及定理 3.7.4 也提供了算法 HIA 收敛速度估计方法. 另外对于算法 IHIA 是否有类似于定理 3.9.2 的结论, 需作进一步研究. 在此仅从定理 3.8.1 及定理 3.8.2 出发给出算法 HIA 收敛速度估计, 即

定理 3.9.3 对于任意初始分布, 算法 IHIA 按几何级数比 $1 - \hat{\delta}$ 收敛.

第十节 免疫算法稳定性理论

在这一部分,首先给出区间函数值理论的基本性质,然后给出一般随机搜索智能算法稳定性的概念,进而探讨免疫算法的稳定性.

一、区间适应值函数的基本性质

设目标函数 $f(x)$ 经扰动后为 $F(x) = f(x) + Z, x \in D, D \subset R^p, Z$ 为随机变量, $Z \in [-a, a], a$ 为给定的正数, 则 $|E[Z]| \leq a$. 由于 $F(x)$ 的取值具有随机性, 因此称 $F(x)$ 为区间适应值函数, $F(x) \in [f(x) - a, f(x) + a]$, 这种区间适应值函数的极小化问题可简约描述为

$$(NBP)' \quad \min_{x \in D} F(x).$$

现引入区间集上偏序关系如下^[7]:

设 $\hat{R} = \{x = [x_1, x_2] \subset \mathbb{R} : -\infty < x_1 \leq x_2 < +\infty\}$ 为区间集合, 对于 $x = [x_1, x_2], y = [y_1, y_2]$, 定义

$$x < y \Leftrightarrow x_2 < y_1; \quad x = y \Leftrightarrow x_i = y_i, i = 1, 2; \quad x \leq y \Leftrightarrow x < y \vee x = y.$$

易于验证 (\hat{R}, \leq) 为偏序集, $(\hat{R}, <)$ 为严格偏序集. 显然 $x < y \Leftrightarrow x \leq y \wedge x \neq y$.

若在 \hat{R} 上定义拓扑结构

$$x \oplus y = [x_1 + y_1, x_2 + y_2], k \circ x = [kx_1, kx_2], k \in \mathbb{R},$$

则 (\hat{R}, \oplus, \circ) 为二维线性空间, 其中一基为 $\{[1, 0], [0, 1]\}$; 若在 (\hat{R}, \oplus, \circ) 上定义范数

$$\| [x_1, x_2] \| = |x_1 - x_2|,$$

则易于验证 $(\hat{R}, \oplus, \circ, \| \cdot \|)$ 为有限维赋范线性空间. 关于这一空间的性质有待进一步研究. 以下考虑扰动函数 $F(x)$ 的比较关系, 引入如下假设条件, 这种条件在以下的稳定性理论结果中不再重述.

基本假设 (H)

- (1) $f(x)$ 在 D 上的最小值存在;
- (2) 免疫算法每迭代一次, $F(x)$ 在 D 仅评价一次.

于是 $F(x)$ 在 D 上的值之间的比较关系可定义如下^[7]:

$$F(x) < F(y) \Leftrightarrow [F(x) - a, F(x) + a] < [F(y) - a, F(y) + a].$$

用 \hat{F} 表示 $F(x)$ 在 D 上已被评价的值构成的集合, 即

$$\hat{F} \equiv F(D) = \{F(x) : x \in D\},$$

引入记号

$$\tilde{f}^* = \inf\{\tilde{f} \in \tilde{F}\}, f^* = \min f(D).$$

现给予问题 (NBP)' 的有效解概念.

定义 3.10.1 称 $x^* \in D$ 为问题 (NBP)' 的有效解, 若不存在 $x' \in D$, 使得

$$F(x') \prec F(x^*).$$

此时简记 $x' \prec x^*$.

此概念与多目标优化的有效解概念的提法相类似, 不过此处是针对区间适应值函数而言. 用 $M_f(F, \prec)$ 表示问题 (NBP)' 的有效解的区间适应值构成的集合, 即

$$M_f(F, \prec) = \{\tilde{f} \in \tilde{F} : \exists \tilde{f}' \in \tilde{F}, \text{ s.t. } \tilde{f}' \prec \tilde{f}\}.$$

根据有效解的定义, 容易获得

$$M_f(F, \prec) = \{\tilde{f} \in \tilde{F} : |\tilde{f} - \tilde{f}^*| \leq 2a, \tilde{f} \in M_f(\tilde{F}, \prec)\}. \quad (3.10.1)$$

以下介绍区间适应值函数的一些性质, 其中定理 3.10.1~3.10.3 分别是对文献 [7] 的定理 1~3 的延拓.

定理 3.10.1 问题 (NBP) 的最优解必是问题 (NBP)' 的有效解, 反之不然.

证明 设 $x^* \in D$ 为问题 (NBP) 的最优解, 则有

$$F(x^*) = f(x^*) + Z \leq f^* + a. \quad (3.10.2)$$

另外, 由 \tilde{f}^* 的定义, 对于 $\forall \varepsilon > 0$, 存在 $\tilde{f} \in \tilde{F}$, 使得

$$\tilde{f}^* + \varepsilon > \tilde{f} = f(x) + Z \geq f^* - a.$$

于是由 ε 的任意性及式 (3.10.2) 可知

$$F(x^*) \leq \tilde{f}^* + 2a,$$

从而由式 (3.10.1) 得到 $F(x^*) \in M_f(F, \prec)$. 其次, 由于随机因素的影响, 显然问题 (NBP)' 的有效解不一定是问题 (NBP) 的最优解. 证毕.

定理 3.10.2 \tilde{f}^* 及 f^* 具有以下关系:

$$|\tilde{f}^* - f^*| \leq a.$$

证明 对于 $\forall x \in D$, 下式成立

$$\tilde{f}^* \leq F(x) = f(x) + Z \leq f(x) + a,$$

则有

$$\tilde{f}^* \leq f^* + a. \quad (3.10.3)$$

另外, 易知

$$F(x) = f(x) + Z \geq f^* - a,$$

则推出

$$\tilde{f}^* \geq f^* - a. \quad (3.10.4)$$

结合式 (3.10.3)、(3.10.4) 可知结论成立.

定理 3.10.3 $\sup\{|\tilde{f} - f^*| : \tilde{f} \in M_f(\tilde{F}, \prec)\} \leq 3a.$

证明 由上确界的定义, 对于 $\forall \varepsilon > 0$, $\exists \tilde{f} \in M_f(\tilde{F}, \prec)$, 使得

$$|\tilde{f} - f^*| + \varepsilon > \sup\{|\tilde{f} - f^*| : \tilde{f} \in M_f(\tilde{F}, \prec)\}. \quad (3.10.5)$$

由式 (3.10.1) 及定理 3.10.2 可推出

$$|\tilde{f} - f^*| < |\tilde{f} - \tilde{f}^*| + |\tilde{f}^* - f^*| \leq 3a, \quad (3.10.6)$$

故由式 (3.10.5)、(3.10.6) 及 ε 的任意性知结论成立. 证毕.

评注 3.10.1 定理 3.10.2 说明 $f(x)$ 在 D 上的最小值 f^* 与在任何时刻 $F(x)$ 在 D 上的下确界 \tilde{f}^* 的偏差始终不超过 a . 定理 3.10.3 说明任何时刻问题 (NBP)' 的有效解的区间适应值与无扰动的问题 (NBP) 的目标函数 $f(x)$ 在 D 上的最小值的偏差不超过 $3a$. 值得一提的是, 在不同时刻, $F(x)$ 在 D 上的下确界 \tilde{f}^* 一般不同.

二、稳定性概念

对于抗体空间 S^N , 在假设 (H) 下, 免疫算法在 $D \equiv S$ 上作用于扰动函数 $F(x)$, 获随机搜索序列 $\{Z[n], n \geq 0\}$, $Z[n] = \min\{F(x) : x \in M(A_n, \prec)\}$, 其中 $M(A_n, \prec)$ 表示在以 A_n 为论域的前提下, A_n 中所有有效个体 (针对 A_n 而言) 构成的集合, 即

$$M(A_n, \prec) = \{x \in A_n : \exists x' \in A_n, \text{s.t. } x' \prec x\}$$

用 $M(S, \prec)$ 表示 $F(x)$ 在 S 上的有效解构成的全体. 为叙述方便, 引入记号

$$\tilde{f}_n^* \equiv \min\{F(x), x \in A_n\}, f_n^* \equiv \min\{\tilde{f} \in F(S_n)\},$$

$$M_f(S, \prec) = \{F(x) | x \in M(S, \prec)\},$$

其中 $F(S_n)$ 表示 n 时刻 $F(x)$ 在 S 上元素的区间适应值的集合.

评注 3.10.2 $Z[n] = \tilde{f}_n^*.$

证明 设 $x_{n0}^* \in A_n, \tilde{f}_n^* = F(x_{n0}^*)$, 若 $x_{n0}^* \notin M(A_n, <)$, 则存在 $x'_n \in A_n$ s.t. $F(x'_n) < F(x_{n0}^*)$, 因此

$$\tilde{f}_n^* + a \leq F(x'_n) + a < F(x_{n0}^*) - a = \tilde{f}_n^* - a.$$

此式产生矛盾, 故 $x_{n0}^* \in M(A_n, <)$. 证毕.

定理 3.10.4 若 $M(A_n, <) \cap M(S, <) \neq \emptyset$, 则

$$Z(A_n) \equiv \min\{F(x) \in F(A_n)\} \in M(A_n, <) \cap M_f(S, <).$$

证明 若 $Z(A_n) \notin M_f(S, <)$, 则对 $\forall x_n^* \in M(A_n, <) \cap M(S, <)$, 必有 $Z(A) < F(x_n^*)$. 其次根据有效解的定义, 存在 $x_0 \in M(S, <)$ 使得 $F(x_0) < Z(A_n)$, 从而

$$F(x_0) + a < Z(A_n) - a < F(x_n^*) - a.$$

这表明 $x_n^* \notin M(S, <)$, 导致矛盾, 故 $Z(A) \in M_f(S, <)$. 另外, 若 $Z(A_n) \notin M(A_n, <)$, 则存在 $y_n \in M(A_n, <)$, 使得

$$F(y_n) < F(y_n) + a < Z(A_n) - a < Z(A_n).$$

这与 $Z(A_n)$ 的定义矛盾. 证毕.

智能算法的稳定性可通过两个随机序列 $\{z[n], n \geq 0\}$ 及 $\{Z[n], n \geq 0\}$ 之间的渐近逼近行为加以描述, 其中, $\{z[n], n \geq 0\}$ 为这种算法独立作用于问题 (NBP) 获得的随机序列, 于是有下面的定义.

定义 3.10.2 在假设 (H) 下, 启发式随机搜索算法称为稳定的, 若其独立分别作用于问题 (NBP) 及问题 (NBP)' 所获得的随机序列 $\{z[n], n \geq 0\}$ 及 $\{Z[n], n \geq 0\}$ 满足: (1) 该算法对问题 (NBP) 概率弱收敛; (2) 对于 $\forall \varepsilon > 0, \exists \delta(\varepsilon) > 0, N(\varepsilon) > 0$, 当时 $a < \delta, n > N(\varepsilon)$ 时, 以下关系成立:

$$E[|Z[n] - z[n]|] < \varepsilon. \quad (3.10.7)$$

定义 3.10.3 在假设 (H) 下, 启发式随机搜索算法称为渐渐稳定的, 若该算法对于问题 (NBP) 概率弱收敛, 且其作用于问题 (NBP)' 所获得的随机序列 $\{Z[n], n \geq 0\}$ 满足: 对于 $\forall \varepsilon > 0, \exists \delta(\varepsilon) > 0, N(\varepsilon) > 0$, 当 $a < \delta, n > N(\varepsilon)$ 时, 以下关系成立:

$$E[|Z[n+1] - Z[n]|] < \varepsilon. \quad (3.10.8)$$

为便于以下定理叙述, 记

$$F_n = A_n \cap M(S, <), \quad G_n = A_{n+1} \cap M(S, <). \quad (3.10.9)$$

三、齐次免疫算法稳定性

利用免疫算法中基本免疫算子的性质, 对于齐次情形, 可得到类似于定理 3.7.4 中式 (3.7.10) 的结论, 即

定理 3.10.5 在假设 (H) 下, 对于问题 (NBP)', 算法 HIA 具有如下性质:

$$P(F_n = \phi) \leq (1 - \delta)^n P(F_0 = \phi). \quad (3.10.10)$$

证明 记

$$\begin{aligned} P_{nX} &= P(G_n = \phi | A_n = X, \quad X \cap M(S_n, \prec) = \phi), \\ \tilde{P}_{nX} &= P(G_n = \phi | A_n = X, \quad X \cap M(S_n, \prec) \neq \phi). \end{aligned}$$

若 $F_n \neq \phi$, 由评注 3.10.2 及定理 3.10.4 知 $Z[n] \in M_f(A_n, \prec) \cap M_f(S, \prec)$. 从而结合最优保存策略得到 $G_n \neq \phi$, 进而

$$P(G_n = \phi | F_n \neq \phi) = 0. \quad (3.10.11)$$

另一方面, 将定理 3.7.3 的证明中 M^* 换为 $M(S, \prec)$, 并类似于式 (3.7.8) 的推导得

$$\begin{aligned} m_n &\equiv \sum_{X \in S^N} P_{nX} \cdot P(A_n = X, \quad X \cap M(S, \prec) = \phi) \\ &\leq (1 - \delta) P(F_n = \phi). \end{aligned} \quad (3.10.12)$$

于是由式 (3.10.11)、(3.10.12) 及全概率公式可导出

$$\begin{aligned} P(G_n = \phi) &= m_n + \sum_{X \in S^N} \tilde{P}_{nX} \cdot P(A_n = X, \quad X \cap M(S, \prec) \neq \phi) \\ &\leq (1 - \delta) P(F_n = \phi). \end{aligned} \quad (3.10.13)$$

另外结合有效解的定义, 可获得

$$P(G_n = \phi) = P(F_{n+1} = \phi), \quad (3.10.14)$$

从而由式 (3.10.13)、(3.10.14) 通过归纳可获结论成立. 证毕.

定理 3.10.6 对于任意初始分布, 算法 HIA 是稳定的, 且满足

$$E[|Z[n] - z[n]|] \leq 3a + M_0(1 - \delta)^n, \quad (3.10.15)$$

其中

$$M_0 \equiv (2 - \delta)(f_{\max} - f^*) + a + E[Z].$$

证明 由定理 3.10.5 的证明知, 若 $F_n \neq \phi$, 则 $Z[n] \in M_f(S, <)$, 进而由最优保存策略知, $Z[n+1] \in M_f(S, <)$, 从而由式 (3.10.1) 可知

$$E[|Z(n+1) - f_n^*| | F_n \neq \phi] \leq 2a. \quad (3.10.16)$$

另一方面, 由于

$$Z[n+1] = \min(F(x_{n+1i}), x_{n+1i} \in A_{n+1}),$$

而

$$F(x_{n+1i}) = f(x_{n+1i}) + Z_{n+1i},$$

其中 $Z_{n+11}, Z_{n+12}, \dots, Z_{n+1q_n}$ 为与随机变量 Z 具有相同分布的独立的随机变量, 因此必有

$$E[Z[n+1]] \leq f_{\max} + E[Z]. \quad (3.10.17)$$

由式 (3.10.16)、(3.10.17) 可导出

$$\begin{aligned} E[|Z[n+1] - f_n^*|] &= E[|Z[n+1] - f_n^*| | F_n = \phi] P(F_n = \phi) \\ &\quad + E[|Z[n+1] - f_n^*| | F_n \neq \phi] P(F_n \neq \phi) \\ &\leq (f_{\max} - f^* + E[Z] + a)(1 - \delta)^n + 2a. \end{aligned} \quad (3.10.18)$$

又由定理 3.7.4 知

$$E[|z[n] - f^*|] \leq (f_{\max} - f^*)(1 - \delta)^n, \quad (3.10.19)$$

从而由定理 3.10.2 及式 (3.10.18)、(3.10.19),

$$\begin{aligned} E[|Z[n+1] - z[n+1]|] &\leq E[|Z[n+1] - f_n^*|] + E[|z[n+1] - f^*|] \\ &\quad + E[|f_n^* - f^*|] \leq 3a + M_0(1 - \delta)^n. \end{aligned} \quad (3.10.20)$$

于是, 式 (3.10.15) 成立. 因而对 $\forall \varepsilon > 0$, $\exists \delta(\varepsilon) = \frac{\varepsilon}{6}$, $N(\varepsilon) = \left\lceil \frac{\ln \varepsilon - \ln(2M_0)}{\ln(1 - \delta)} \right\rceil$, 当 $a < \delta(\varepsilon)$, $n > N(\varepsilon)$ 时,

$$E[|Z[n] - z[n]|] < \varepsilon,$$

故由定义 3.10.2 知该算法是稳定的. 证毕.

定理 3.10.7 对于任意初始分布, 算法 HIA 是渐渐稳定的.

证明 当 $F_n \neq \phi$ 时, 由定理 3.10.6 的证明可以看出, $Z[n]$ 及 $Z[n+1]$ 皆属于 $M_f(S_n, <)$, 于是由式 (3.10.1) 有

$$|Z[n+1] - Z[n]| \leq 4a. \quad (3.10.21)$$

进而结合式 (3.10.10) 及

$$E[|Z[n+1] - Z[n]|] = E[|Z[n+1] - Z[n]| \mid F_n = \phi]P(F_n = \phi) + E[|Z[n+1] - Z[n]| \mid F_n \neq \phi]P(F_n \neq \phi)$$

可推出

$$E[|Z[n+1] - Z[n]|] \leq (f_{\max} - f^* + 2a)(1 - \delta)^n + 4a. \quad (3.10.22)$$

进而对于 $\forall \varepsilon > 0$, $\exists \delta(\varepsilon) = \frac{\varepsilon}{8}$, $N(\varepsilon) = \left\lceil \frac{\ln \varepsilon - \ln(2(f_{\max} - f^* + 2a))}{\ln(1 - \delta)} \right\rceil$, 当 $a < \delta(\varepsilon)$, $n > N(\varepsilon)$ 时, 下式成立:

$$E[|Z[n+1] - Z[n]|] < \varepsilon,$$

故由定义 3.10.3 知该定理成立. 证毕.

为了从仿真的角度验证所获稳定性理论结果, 选取随机变量 $Z = Z_1 * Z_2$, $Z_i, i = 1, 2$ 在 $[-a, a]$ 上服从均匀分布, $a = 1$, 这样选取的目的在于检测算法 HIA 抗干扰的能力.

将算法 HIA 分别独立用于问题 (NBP) 及问题 (NBP)', 对同一测试函数所获独立随机序列 $\{z[n], n \geq 0\}$ 及 $\{Z[n], n \geq 0\}$ 所对应的结果放入同一图形中, 其目的是便于比较. 在此仅以例 2.5.2 中函数 F_1 及 F_4 为例加以说明. 算法 HIA 用于问题 (NBP)' 所获进化群体的最大值随机序列与其无扰动的问题 (NBP) 的最大值的偏差用序列 $\{error[n], n \geq 0\}$ 表示, $error[n] = |f_{\max} - Z[n]|$, 从此序列可反映出序列 $\{Z[n], n \geq 0\}$ 与理想解的目标值的偏离程度. 该算法作用于问题 (NBP) 及问题 (NBP)' 的一次实验结果如图 3-5~3-7 所示.

由图 3-5(1) 及图 3-7(1) 可看出, 算法 HIA 在处理极大化问题时, 随着进化代数的增大, 噪声的影响越来越小, 且群体中最大区间适应值在理想值的附近作微小变动, 并且满足 $f_{\max} \leq Z[n] \leq f_{\max} + a$, 此时 $Z[n]$ 表示 $F(x)$ 在 A_n 上的最大值, 这正好符合期望的目标. 另外, 由图 3-5(2) 及图 3-7(2) 可知, 随着迭代次数的

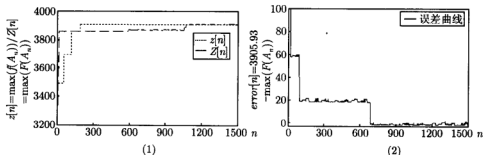


图 3-5 函数为 F_1 : (1) 进化群体的最大值序列比较; (2) 用于问题 (P)' 所获进化群体的最大值序列与理想值的误差序列比较

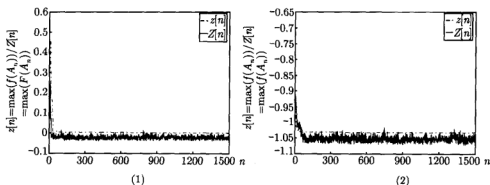


图 3-6 (1) 函数 F_1 : 进化群体最小值序列比较; (2) 函数 F_4 : 进化群体最小值序列比较

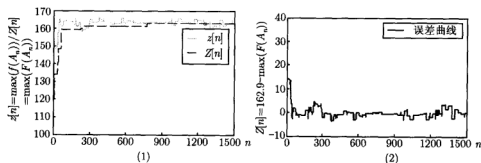


图 3-7 函数 F_4 : (1) 进化群体的最大值序列比较; (2) 用于问题 (P)' 所获进化群体的最大值序列与理想值的误差序列比较

增加, 区间适应度函数在进化群体上的最大值与理想解的目标值的偏差逐步减小, 这反映了噪声扰动不会使算法的搜索出现较大幅度的波动。

其次, 由图 3-6 可看出, 算法 HIA 在处理含噪声干扰的极小化问题时, 群体搜索逐渐使当前群体的最小区间适应度值逐渐减小, 并到了一定的迭代次数后, 群体中区间适应值的最小值保持在衡定的较小范围变化, 即 $f_{\min} - a \leq Z[n] \leq f_{\min}$ 。

综上所述, 对于任意随机变量的扰动, 当幅度值 a 取值较小时, 算法 HIA 的搜索不会因扰动的影响而出现较大幅度的跳跃; 当 a 较大时, 偏离稍大。这说明噪声的幅值 a 是确定序列 $\{Z(n), n \geq 0\}$ 偏离序列 $\{z[n], n \geq 0\}$ 的重要因素。所获理论结果与实验结果一致, 因而该算法具有较好的稳定性。

四、非齐次免疫算法稳定性

现给出如下定理。

定理 3.10.8 对于任意初始分布, 算法 IHIA 是稳定的, 且满足

$$E[|Z[n] - z[n]|] \leq M_1(1 - \delta)^n + 3a, \quad (3.10.23)$$

其中 b 由定理 3.8.2 确定, $M_1 = b(2(f_{\max} - f^*) + a)$.

证明 为叙述方便, 引用定理 3.10.5 中的记号 P_{nX}, \bar{P}_{nX} . 当 $G_n \neq \phi$ 时, 必有 $Z[n+1] \in M_f(S, \prec)$. 否则必导致:

- (1) 存在 $\bar{x}_{n+1}^* \notin M(S, \prec)$, 使得 $Z[n+1] = F(\bar{x}_{n+1}^*)$;
- (2) $Z[n+1] < F(x_{n+1}^*), \forall x_{n+1}^* \in A_{n+1} \cap M(S, \prec)$, 从而存在 $x'_{n+1} \in S$, 使得

$$f_n^* + a \leq F(x'_{n+1}) + a < Z[n+1] - a < F(x_{n+1}^*) - a.$$

即有

$$F(x_{n+1}^*) > f_n^* + 2a.$$

这与 $x_{n+1}^* \in M(S, \prec)$ 相矛盾, 故由式 (3.10.1) 可知

$$E(|Z[n+1] - f_n^*| \mid G_n \neq \phi) \leq 2a. \quad (3.10.24)$$

进而由定理 3.8.2、定理 3.10.2 及式 (3.10.24) 导出

$$\begin{aligned} E[|Z[n+1] - z[n+1]| \mid G_n \neq \phi] &\leq E[|Z[n+1] - f_n^*| \mid G_n \neq \phi] \\ &\quad + E[|z[n+1] - f_n^*|] + E[|f_n^* - f^*|] \\ &\leq 3a + b(f_{\max} - f^*)(1 - \hat{\delta})^n. \end{aligned} \quad (3.10.25)$$

其次, 将 $M(S, \prec)$ 视为 M^* , 并类似于定理 3.8.1 中式 (3.8.6) 及式 (3.8.10) 的推导, 可产生下列式子:

$$P(G_n = \phi \mid F_n \neq \phi) \leq \beta_n,$$

$$P(G_n = \phi \mid F_n = \phi) \leq 1 - \hat{\delta}.$$

于是, 由全概率公式有

$$P(G_n = \phi) \leq (1 - \hat{\delta})P(F_n = \phi) + \beta_n. \quad (3.10.26)$$

又因下式成立:

$$P(G_n = \phi) = P(F_{n+1} = \phi), \quad (3.10.27)$$

结合式 (3.10.26)、(3.10.27), 并通过归纳导出

$$P(G_n = \phi) \leq b(1 - \hat{\delta})^{n+1}, \quad (3.10.28)$$

从而结合式 (3.10.25) 及 (3.10.28) 可导出

$$\begin{aligned} E[|Z[n+1] - z[n+1]|] &\leq E[|Z[n+1] - z[n+1]| \mid G_n \neq \phi]P(G_n \neq \phi) \\ &\quad + E[|Z[n+1] - z[n+1]| \mid G_n = \phi]P(G_n = \phi) \\ &\leq 3a + b(f_{\max} - f^*)(1 - \hat{\delta})^n + (f_{\max} - f^* + a)b(1 - \hat{\delta})^n \\ &= M_1(1 - \hat{\delta})^n + 3a. \end{aligned}$$

故式 (3.10.23) 成立, 进而由稳定性的定义知该算法是稳定的. 证毕.

定理 3.10.9 对于任意初始分布, 算法 IHIA 是渐渐稳定的.

证明 当 $G_n \neq \phi, F_n \neq \phi$ 时, 由定理 3.10.7 的前半部分证明可看出, $Z[n]$ 及 $Z[n+1]$ 皆属于 $M_f(S, <)$, 从而由式 (3.10.1) 有

$$|Z[n+1] - Z[n]| \leq 4a.$$

当 $G_n \neq \phi, F_n = \phi$ 时, 结合式 (3.10.28) 及下式:

$$\begin{aligned} & E[|Z[n+1] - Z[n]| \mid G_n \neq \phi] \\ & \leq E[|Z[n+1] - Z[n]| \mid G_n \neq \phi, F_n \neq \phi] P(F_n \neq \phi) \\ & \quad + E[|Z[n+1] - Z[n]| \mid G_n \neq \phi, F_n = \phi] P(F_n = \phi) \\ & \leq 4a + (f_{\max} - f^* + 2a)b(1 - \hat{\delta})^n, \end{aligned} \quad (3.10.29)$$

从而由式 (3.10.28)、(3.10.29) 及下式获知:

$$\begin{aligned} & E[|Z[n+1] - Z[n]|] = E[|Z[n+1] - Z[n]| \mid G_n = \phi] P(G_n = \phi) \\ & \quad + E[|Z[n+1] - Z[n]| \mid G_n \neq \phi] P(G_n \neq \phi) \\ & \leq (f_{\max} - f^* + 2a)b(1 - \hat{\delta})^{n+1} + 4a + (f_{\max} - f^* + 2a)b(1 - \hat{\delta})^n \\ & = b(f_{\max} - f^* + 2a)(2 - \hat{\delta})(1 - \hat{\delta})^n + 4a \end{aligned}$$

故由定义 3.10.3 知该定理成立. 证毕.

类似于算法 HIA 的稳定性仿真实例分析的方法, 同样利用测试函数 F_1 及 F_4 对所获算法 IHIA 稳定性理论结果进行验证, 随机变量 Z 及幅度值 a 的选择、随机序列 $\{z[n], n \geq 0\}$ 、 $\{Z[n], n \geq 0\}$ 及 $\{error[n], n \geq 0\}$ 的含义以及图形绘制与第三章第十节中仿真部分的叙述相同. 实验结果如图 3-8~3-10 所示.

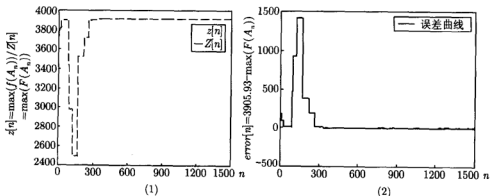
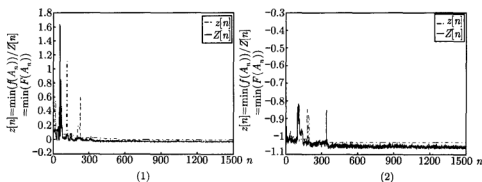
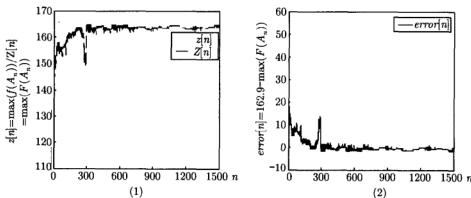


图 3-8 函数为 F_1 : (1) 进化群体的最大值序列比较; (2) 用于问题 (NBP)' 所获进化群体的最大值序列与理想值的误差序列比较


 图 3-9 (1) 函数为 F_1 : 进化群体最小值序列比较; (2) 函数为 F_4 : 进化群体最小值序列比较

 图 3-10 函数为 F_4 : (1) 进化群体的最大值序列比较; (2) 用于问题 (NBP)' 所获进化群体的最大值序列与理想值的误差序列比较

从图 3-8(1) 及图 3-10(1) 看, 该算法用于问题 (NBP)' 时, 随迭代数的增加, 当前群体的最大值在 $[f_{\max}, f_{\max} + a]$ 范围变化; 从图 3-8(2) 及图 3-10(2) 看, 最大值序列与理想值的偏差序列 $\{error[n], n \geq 0\}$ 随迭代数的增加, 其变化逐渐微小; 由图 3-9 观察, 该算法用于问题 (NBP)' 所获进化群体的最小值序列逐渐在 $[f_{\min} - a, f_{\min}]$ 范围变化, 并且不会因扰动的影响而出现所获进化群体的最小值的变化幅度较大的现象. 以上实验结果与该算法稳定性理论结果一致, 因而此算法具有较好的稳定性.

第十一节 免疫算法的计算复杂度及鲁棒性分析

一、计算复杂度分析

算法中包含四个参数, 即群体规模 N 、克隆群大小 M 、浓度抑制半径 σ 、自我抗体产生比率 μ . 在第 n 次搜索过程中, 突变克隆群的亲和度计算次数不超过

M , 此群体的浓度计算次数为 $\frac{1}{2}M(M-1)$, 参与免疫选择的群体的浓度计算次数为 $\frac{1}{2}N(N-1)$, 随机产生的自我抗体群的亲和力计算次数不超过 μN , 于是第 n 次迭代中算法计算的总次数 P_n 满足

$$\begin{aligned} P_n &\leq M + \frac{1}{2}M(M-1) + \frac{1}{2}N(N-1) + \mu N \\ &= \frac{1}{2}(M^2 + N^2 + M - (2\mu - 1)N), \end{aligned}$$

由此推出免疫算法的计算最高级为 $O(M^2)$. 这表明克隆群规模及浓度计算对算法搜索速度有直接影响.

二、免疫算法鲁棒性分析

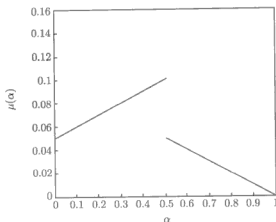
免疫算法的鲁棒性分析是测试算法对参数敏感程度的重要途径. 从算法的描述可看出, 在给定群体规模 N 和细胞克隆数 M 的条件下, 该算法包含三个关键参数 α, σ, μ , 其余参数皆由 α 或 μ 表示, 即

$$N_1 = \alpha \cdot N, \quad N_2 = (1 - \mu) \cdot N, \quad N_3 = \mu \cdot N$$

这三个参数 α, σ, μ 对该算法的搜索性能都有不同程度的影响. 在此以例 2.5.2 中函数 F_3 为代表对免疫算法 (包含齐次及非齐次情形) 鲁棒性进行测试. 由图 2-7 可看出, 此函数的性质较差, 一般的智能算法较难获得理想结果. 通常 α 与 μ 成反比关系, 为便于从图形观察, 对于免疫算法, 规定抗体群规模取为 80, 终止代数取为 1500. 在给定的参数组 (α, σ, μ) 及终止迭代次数下, 若算法一次运行所获的最好解与理想解 (即最优解) 的函数值误差不大于 ε , 则称此次试验成功; 在一组参数下, \bar{M} 次试验成功的比率作为在这组参数下的收敛概率, 即 \bar{M} 次试验成功的次数与 \bar{M} 的比. 在此选定 $\varepsilon = 0.05$, 试验次数 \bar{M} 取为 500. 如果在独立运行一次的过程中, 该算法未能找到满足要求精度的解, 则规定此时首达时间为 2000, 意指未找到满足要求的解; 若在规定的迭代数范围内, 已找到满足要求精度的解, 则以到获得满足要求精度的解时刻为止的迭代次数作为首达时间. 平均计算次数意指同一组参数值下, \bar{M} 次试验对目标函数计算次数 (指每次试验直到首次找到满足要求解时为止) 的平均值. 鲁棒性测试考虑两种情形: 一是让 α 与 μ 具有确定的关系 $\mu(\alpha)$, 而 σ 为自由变量, 即选取 α 与 $\mu(\alpha)$ 满足下列关系, 这可由图 3-11 描述

$$\mu(\alpha) = \begin{cases} \frac{0.5 + \alpha}{10}, & 0 \leq \alpha \leq 0.5, \\ \frac{1 - \alpha}{10}, & 0.5 < \alpha \leq 1. \end{cases} \quad (3.11.1)$$

另一种情形是让 σ 不变化, 而 α 及 μ 作为自由变量. 以下首先对算法 HIA 的鲁棒性进行检测, 其次对算法 IHIA 的鲁棒性进行检测.


 图 3-11 函数 $\mu(\alpha)$ 图形

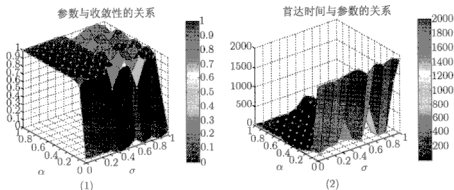
三、齐次免疫算法鲁棒性

利用 Shubert 函数 F_3 对算法 HIA 鲁棒性进行分析。

情形 1 α 与 μ 满足式 (3.11.1), σ 为自由变量, $0 \leq \sigma \leq 1$ 。

图 3-12 及图 3-13 (1) 分别是该算法作用于函数 F_3 所获的 α, σ 与收敛性、首达时间及平均计算次数关系图, 图中的符号“o”表示算法所获数值值所对应的点。对于每一组参数 (α, σ, μ) , 算法独立运行 500 次。

由图 3-12(1) 可知, 收敛区为 $S_1: \{(\alpha, \sigma) | 0.1 < \alpha \leq 1, 0 \leq \sigma \leq 0.5\}$, 很危险区为 $S_2: \{(\alpha, \sigma) | 0 \leq \alpha \leq 0.1, 0 \leq \sigma \leq 1\}$, 危险区为 $S_3: \{(\alpha, \sigma) | 0 \leq \alpha < 0.2, 0.5 < \sigma \leq 1\}$ 。当 $\alpha = 0$ 时, 由式 (3.11.1) 知 $\mu = 0.05$, 此时克隆选择、细胞克隆、亲和突变及克隆抑制算子对该算法无贡献, 不论 σ 为何值, 该算法都不收敛, 仅有免疫选择及募集新成员对群体的搜索有贡献, 搜索完全处于随机状态, 因而算法肯定不能保证收


 图 3-12 情形 1: (1) α, σ 与收敛概率的关系; (2) 首达时间与 α, σ 的关系

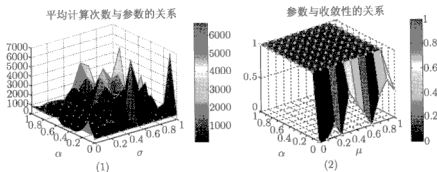


图 3-13 (1) 情形 1: 平均计算次数与 α, σ 的关系; (2) 情形 2: 参数 α, μ 与收敛性的关系

敛性; 当 $\alpha = 1$ 时, 由式 (3.11.1) 知 $\mu = 0$, 若 $0 \leq \sigma \leq 0.5$, 则算法收敛, 这种情形下, 募集新成员对算法无作用, 但也能保证算法的收敛性, 而由图 3-13(1) 可看出, 此时平均计算次数较大, 即为 869 次; 而当 $0.2 \leq \alpha < 1$ 时, μ 不为 0, 此时若 $0 < \sigma \leq 0.5$, 则平均计算次数不超过 750 次, 这说明 μ 对算法搜索性能具有改进作用; 特别在图 3-13(1) 的区域 $\{(\alpha, \sigma) | 0.1 \leq \alpha \leq 0.3, 0.8 \leq \sigma \leq 1\}$ 内出现一块子区域上算法收敛现象, 这表明算法中抑制半径较大, 而 $\mu \in (0.06, 0.08)$, 此时募集新成员的作用较大, 但由图 3-12(2) 及图 3-13(1) 可反映出平均计算次数最小为 90 次, 最大为 3785 次, 首达时间最小为 13, 最大为 541, 此表明当 $\sigma \in [0.8, 1]$ 时, σ 的选取容易影响算法的搜索速度, 因而此收敛区域不可取。

另外对于参数 σ 在 $\sigma > 0.4$ 时, 由图 3-12 及图 3-13 (1) 可看出, σ 对算法的收敛性、首达时间及平均计算次数的影响是相当明显的, 因此为了考虑算法的搜索性能, 参数 α, σ, μ 的选取范围为

$$\{(\alpha, \sigma, \mu) | 0.2 \leq \alpha \leq 0.6, 0 < \sigma \leq 0.4, 0.05 \leq \mu \leq 0.08\},$$

其中 α 与 μ 成反比。

情形 2 $\sigma = 0.4$, α 及 μ 作为自由变量

在 σ 给定, α 及 μ 在 $[0, 1]$ 上不受约束的前提下, 对于给定的每一组参数 (α, μ) , 将算法 HIA 对函数 F_3 运行一次, 可获对于所有参数组下的收敛情况如图 3-13(2) 所示。由此图可观察出, 当 $\alpha \in [0.1, 1], \mu \in [0, 0.9]$ 时, 算法收敛, 当 $\alpha \in [0, 0.1], \mu \in [0, 0.1]$ 时, 算法不能被保证收敛。特别当 $\mu = 0, \alpha \geq 0.1$ 时, 能获收敛性, 但是到收敛时刻所进行的平均计算次数最大为 5410, 最小为 183, 这表明计算代价较高, 因此应考虑选取 $\mu > 0$ 。对于 $\alpha \geq 0.1$ 时, μ 的选取不影响算法的收敛性, 但 μ 对算法的搜索性能有一定的促进作用, 而 μ 若与 α 成正比, 或若 α 过小, μ 过大, 皆会影响搜索速度, 因而通常选取若 α 过小, 则 μ 稍大, 即接近 0.08。若 α 过大, 则 μ 稍小, 即 μ 靠近 0.05。 α, μ 的最佳范围是

$$\alpha \in [0.2, 0.6], \mu \in [0.05, 0.08].$$

通过鲁棒性分析获知, μ 的选取范围与免疫系统自适应产生新抗体的机制相一致。

四、非齐次免疫算法鲁棒性

算法 IHIA 鲁棒性的分析方式与齐次情形相似。由于算法 HIA 与算法 IHIA 对参数的要求有所不同, 故在此节讨论非齐次免疫算法鲁棒性问题。

情形 1 α 与 μ 满足式 (3.11.1), σ 为自由变量

对于每一组参数 (α, σ) , 算法 IHIA 对函数 F_3 运行一次, 终止代数数为 1500。在各组参数下, 所获收敛性、首达时间及平均计算次数情况分别如图 3-14 及 3-15(1) 所示。

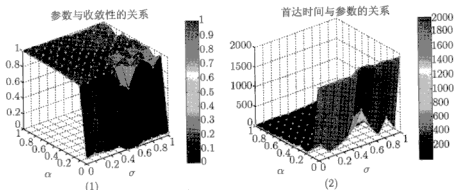


图 3-14 (1) α, σ 与收敛概率的关系; (2) 首达时间与 α, σ 的关系

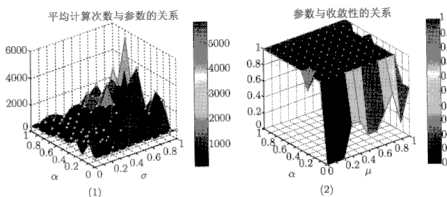


图 3-15 (1) 情形 1: 平均计算次数与 α, σ 的关系; (2) 情形 2: 参数 α, μ 与收敛性的关系

由图 3-14(1) 可知, 收敛区域为 $S_1: \{(\alpha, \sigma) | 0.2 \leq \alpha \leq 1, 0 \leq \sigma \leq 0.5\}$, 很危险区域为 $S_2: \{(\alpha, \sigma) | 0 \leq \alpha \leq 0.4, 0.5 \leq \sigma \leq 1\}$, 有较小危险的区域为 $S_3: \{(\alpha, \sigma) | 0.4 \leq \alpha \leq 1, 0.5 \leq \sigma \leq 1\}$, 同时区域 $S_1 \cup S_3$ 非常平坦, 如果迭代数很大, 也

该算法在区域 S_3 上也能获收敛性,但在区域 S_2 上由于该算法在搜索过程中,当前群体被选择克隆的抗体数较少,而抑制半径较大,导致较好的抗体被保存到下一代的机会较小,进而出现较多不收敛现象,因而此区域上的参数不可取;从区域 $S_1 \cup S_3$ 可反映出,该算法对参数的敏感度低;对 $\alpha=0$ 及 $\alpha=1$ 的情形所获结论与齐次情形相同;从首达时间整体看(见图 3-14(2)),该算法找到满足要求的解的首达时间最小为 6,最大为 554;从平均计算次数看(见图 3-15(1)),在收敛或接近收敛情形,最小平均计算次数为 278,最大平均计算次数为 2868 次,这表明该算法有一定的计算复杂度,原因在于此算法在搜索初期,群体搜索的范围较广。

情形 2 $\sigma=0.45$, α 及 μ 作为自由变量

此情形的 α, μ 的选取与齐次情形的情形 2 相似,并且所获结果也相类似,如图 3-15(2) 所示. 与齐次情形的情形 2 不同的是,该算法在 $\alpha=0.1, \mu=0$ 时,不能保证收敛性;其次,在 $\alpha>0.1, \mu=0$ 时,到收敛时刻的平均计算次数皆大于 395,此比齐次情形要高; α, μ 的最佳范围与齐次情形相同。

第十二节 齐次及非齐次免疫算法理论比较分析

为了获知算法 HIA 及算法 IHIA 的区别,本节从算法设计、鲁棒性、收敛性、收敛速度及稳定性角度分析此两算法的区别与联系. 记算法 HIA 为算法 A, 算法 IHIA 为算法 B.

一、算法设计

算法 A 与算法 B 所包含的基本算子中仅存在免疫选择算子不同,但这两种算法却从不同角度反映马氏链的一些特性. 对于算法 B, 细胞克隆数目、亲和突变率及克隆抑制半径均可设计与迭代数有关,使得这些算子与时间有关,提高算法的搜索速度. 这两种算法都存在较好的群体多样性,这可由图 3-16~3-23(2) 获知.

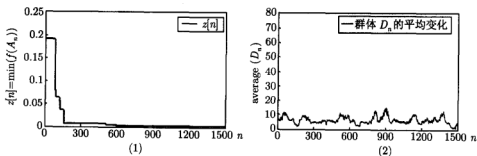
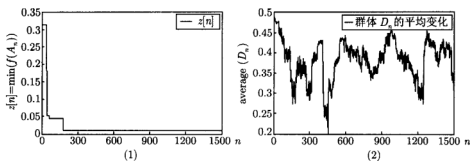
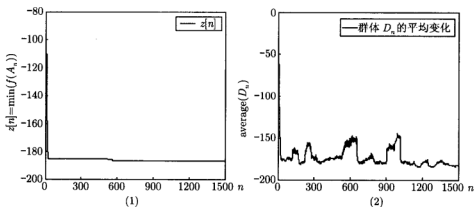
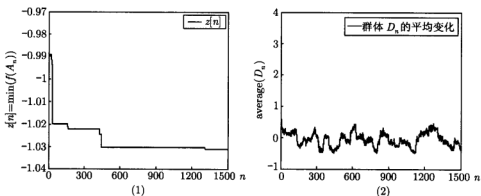


图 3-16 函数 F_1 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

图 3-17 函数 F_2 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线图 3-18 函数 F_3 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线图 3-19 函数 F_4 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

二、鲁棒性、搜索速度及计算复杂度

参数灵敏度是衡量算法性能的重要标志。从第三章第十一节获知, 算法 B 比

算法 A 对参数的敏感度要低,在参数所在的较大区域几乎能获收敛性,而算法 A 对于参数 $\sigma \geq 0.5$ 却表现出较强的敏感性,特别在 $\sigma > 0.5$ 时,表现较突出。

从搜索速度看,算法 A 获满足要求的解比算法 B 快,且在迭代数小的时候,算法 B 所获当前进化群体的较好解不一定被保存,仅当迭代数达到一定的值后所获满足要求的解才被保存,而算法 A 一旦找到满足要求的解后就被保存,原因在于此算法含最优保存策略,此分析可由图 3-16~3-19(1) 反映出。

从搜索效果看,算法 B 比算法 A 好,算法 A 易于仅能获次优解,而算法 B 易于获得整体最优解(如图 3-20~3-23(1)),算法 B 具有突现性质,即当进化的群体陷入局部区域时,算法 B 以概率方式接受稍微差的解,从而能搜索到更好的解。算法 B 找到满足要求的解的首达时间最小为 6,最大为 554。整体上看,算法 B 比算法 A 的首达时间小。

从平均计算次数看,算法 B 在收敛或接近收敛情形,最小平均计算次数为 278,最大平均计算次数为 2868 次,这明显高于算法 A 的收敛或接近收敛的平均计算次数,原因在于该算法在搜索初期,群体搜索的范围较广。

从计算复杂度看,这两种算法在一次迭代过程中,皆需对所有抗体计算浓度,因此计算复杂度为 $O(M^2)$ 。

三、收敛性

第 4.2~4.3 节已从理论上获知算法 A 及算法 B 皆收敛,这收敛结果在图 3-16~3-23(1) 中获得验证;从仿真效果获知,算法 B 所获解的精确度比算法 A 的情形高,但算法 B 要求的终止迭代次数比算法 A 稍高。对于较困难的优化问题,算法 A 易于仅获得次优解。

四、稳定性

从图 3-12~3-15 可看出,算法 B 抗干扰能力明显较算法 A 强;在含扰动的情形下,算法 B 在搜索初期,波动较明显,但经过一定的迭代数后,该算法几乎不受随机扰动项的影响。算法 A 在搜索后期,干扰因素还有微小作用。

从实际应用的角度,经仿真获知,这两算法都具有的共同特征是,在含扰动项的前提下,当随机变量的最大幅值为 \max_z ,最小幅值为 \min_z 时,两种算法若处理极小化问题,则搜索到一定的代数后,所获当前群体的最小值 $f_{n \min}^*$ 满足 $f^* - \min_z \leq f_{n \min}^* \leq f^*$;若处理的问题是极大化问题,则当前群体的最大值 $f_{n \max}^*$ 满足 $f_{\max} \leq f_{n \max}^* \leq f_{\max} + \max_z$,这恰符合期望的目标。另外,从图 3-9(2) 及图 3-10(2) 可看出,算法 B 所获误差较小,在迭代数增大时,误差值接近 0,而算法 A 仍有微小波动。

综上所述, 算法 A 及算法 B 各有其自身的优缺点, 在实际应用中可根据问题的需要而定。一般来说, 选择算法 B 比较合适, 尽管此算法搜索最优解速度比算法 A 稍慢, 但所获得解的精度比算法 A 的情形高。

第十三节 免疫算法的性能测试

免疫算法的参数选定为: 群体规模 N 为 80, 二进制串长 l 为 20, 终止代数 1500, 克隆数 M 为 100, 克隆选择率 α 为 0.6, 抑制半径 σ 为 0.2, 随机产生新抗体的比率 μ 为 0.08, 算法 IHIA 的初始温度为 100。

一、齐次免疫算法的数值实验

算法 HIA 分别作用于例 2.5.2 的函数 $F_1 \sim F_4$ 搜索最小值, 在此仅列举对各测试函数的一次运行结果, 如图 3-16~3-19。图 3-16~3-19 (1) 描述该算法搜索最优解的当前群体 A_n 的最小值 $z[n]$ 变化情况, $z[n]$ 由式 (3.5.3) 确定。该算法对函数 $F_1 \sim F_4$ 所获实际解的目标值为 0.00161408, 0.00999761, -186.594, -1.03128。相应地, 图 3-16~3-19 (2) 分别为获图 3-16~3-19 (1) 的过程中随迭代数的增加, 经免疫选择所获群体 D_n 的目标值 (即抗体对应候选解的函数值) 的平均值 $\text{average}[n]$ 的变化。即

$$\text{average}[n] = \frac{1}{N_2} \sum_{x \in D_n} f(x) \quad (3.13.1)$$

从图 3-16~3-19(1) 看, 不同的测试问题寻优的进程有所不同。对于测试问题 F_1 及 F_2 , 该算法搜索速度快, 在第 300 代左右能搜索到最优解, 但对于函数 F_3 及 F_4 , 却至少需 600 代才能搜索到次优解或最优解。

群体的平均值变化可反映群体搜索的整体行为, 如果群体的平均值随迭代数的增大而不再变化, 则群体不再进化, 并趋于相对平衡状态。从当前群体的平均值变化的一次运行结果 (图 3-16~3-19(2)) 获知, 齐次免疫算法随迭代数的增加, 当前群体的平均值始终不恒同, 这反映了该算法具有较好的群体多样性。

二、非齐次免疫算法的数值实验及比较

(一) 低维多峰值函数优化测试

为了便于与算法 HIA 比较, 在此仍然让算法 IHIA 作用于例 2.5.2 中函数 $F_1 \sim F_4$ 。该算法除测试函数 F_2 仅获目标值为 0.00999761 的次优解外, 对其他测试问题均已获最优解。此算法用于搜索测试函数的最小值的一次运行结果如图 3-20~3-23 所示。

从图 3-20~3-23(1) 可看出, 该算法在搜索初期, 搜索范围较宽, 波动性较大, 原因在于由初始温度及无最优保存策略, 但经过一定的迭代次数后, 搜索范围变狭窄, 并逐步搜索到最优解。特别当算法搜索经过局部区域时, 此算法能通过自适应地促成进化群体跳出此局部区域, 以至达到搜索整体最优解的目的, 这表明该算法具有突现性质。图 3-20~3-23 (2) 反映了该算法搜索群体具有多样性。从搜索效果看, 算法 IHIA 比算法 HIA 所获的效果好。

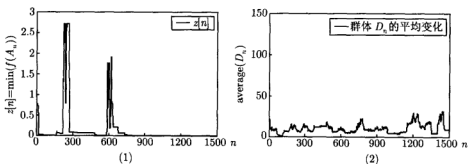


图 3-20 函数 F_1 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

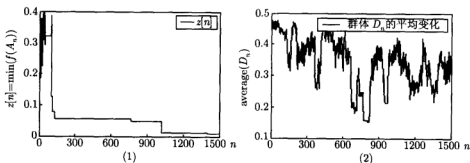


图 3-21 函数 F_2 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

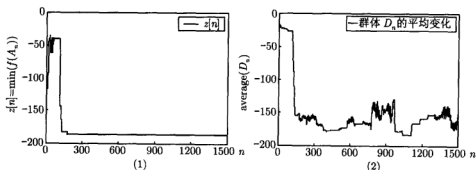


图 3-22 函数 F_3 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

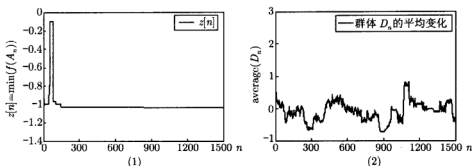


图 3-23 函数 F_4 : (1) 群体 A_n 的最小值变化曲线; (2) 群体 D_n 的平均值变化曲线

(二) 高维函数优化测试

在以下实验中, 参数选取范围是 $0.52 \leq \alpha \leq 0.6, 0.1 \leq \sigma < 0.5, 0.05 \leq \mu \leq 0.08$, 这些参数的具体取值及其余参数取值视优化问题的难易程度而定。

为了获知算法 IHIA 对一般函数优化的搜索性能, 考虑以下测试函数^[17] $f_1 \sim f_5$ 的极大化问题。文献 [17] 利用下列函数对进化策略 (ES)、遗传算法 (GA) 及该文献的免疫遗传算法 (IGA) 进行了测试, 获得的结论是, ES 处理目标函数

$$\begin{aligned}
 f_1(X) &= 10 - \sum_{i=1}^n (x_i - 5)^2, 1 \leq x_i \leq 10, \max f_1(X) = 10, x_i = 5, \\
 f_2(X) &= 6n + \sum_{i=1}^n \text{integer}(x_i), 1 \leq x_i \leq 10.1, \max f_2(X) = 80, x_i = 10, \\
 f_3(X) &= 10 - 100 \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 - \sum_{i=1}^n (x_i - 1)^2, 1 \leq x_i \leq 10, \\
 &\quad \max f_3(X) = 10, x_i = 1, \\
 f_4(X) &= \frac{\sin\left(\sum_{i=1}^n |x_i - 5|\right)}{\sum_{i=1}^n |x_i - 5|}, 1 \leq x_i \leq 10, \max f_4(X) = 1, x_i = 5, \\
 f(X) &= 900 - \sum_{i=1}^n ((x_i - 5)^2 - 10 \cos(2\pi(x_i - 5))), \\
 &\quad 1 \leq x_i \leq 10, \max f_5(X) = 950, 1000, n = 5, 10, x_i = 5
 \end{aligned}$$

为光滑函数的优化问题能获得最好效果, IGA 对多峰值函数及非连续的优化问题能获得较好效果, GA 仅对凸优化问题有较好效果。以下对 IHIA 的性能测试中, 为了提高 IHIA 的搜索速度, 将其亲和突变中的克隆在按与其母体的亲和力成反比的可变概率作用下, 重复突变到满足一定的条件为止, 即克隆 Ab 重复突变为克隆 Ab' , 使 Ab' 的目标值 (即对应的候选解的目标函数值) 满足 (对极大化问题) $f(Ab') \geq$

$f(Ab) - \varepsilon$, 其中 ε 依据优化问题的难易程度而定. 算法 ES、GA 或 IGA 对每一测试问题的第 i 次独立运行所获最好解的目标函数值 f_i^* 若满足 $f_i^* \geq \varepsilon_1$, 则称该算法对该测试问题的此次实验成功. 同理若 IHIA 运行过程中, 能获得当前群体中最好抗体的目标值满足 $f_i^* \geq \varepsilon_2$, 则也称该算法对该测试问题的第 i 次实验成功, 且此时算法运行要求终止. 特别要求 $\varepsilon_1 \leq \varepsilon_2$, 对于函数 f_1 、 f_3 及 5 维的 f_5 , 要求 $\varepsilon_1 < \varepsilon_2$, 其目的是要求 IHIA 搜索最好解的精度比其余三种算法的精度高.

以上四种算法均对以上每一测试问题连续独立运行 30 次, 用 NS 及 AE 分别表示任一种算法对任一测试问题的 30 次独立运行的成功次数及平均计算次数, NF 表示失败次数^[17], NF=30-NS, BV、WV、AV 分别表示此 30 次独立运行结果的最好解的目标值、最差解的目标值、所有解的目标值的平均值. 于是以上算法对测试问题 $f_1 - f_5$ 求最大值的 30 次结果比较如表 3-3~3-7 所示, 其中算法 ES、GA、IGA 的结果选自文献 [17].

函数 f_1 为凸函数, 表 3-3 说明以上四种算法均能搜索到最优解, IGA 的平均计算次数最高, 这表明一般智能算法对凸优化问题均能获得较满意解.

表 3-3 ES、GA、IGA、HIA 对 f_1 30 次运行结果比较

函数 $f_1, \varepsilon_1 = 9.998, \varepsilon_2 = 9.99, n = 5$						
	NF	NS	BV	WV	AV	AE
ES	0	30	10.0	10.0	10.00	5750
GA	0	30	10.0	10.0	10.00	12240
IGA	0	30	10.0	9.998	10.00	22605
IHIA	0	30	10	10	10	13868

函数 f_2 为非连续的阶梯函数, 表 3-4 表明 IGA 比 NS、GA 所获效果好, 特别 IHIA 要求所获最佳解的精度比其余三种算法的精度都高且所获的成功次数仍为 30, 而且平均值比 IGA 的高, 因此对于 f_2 , IHIA 比其余三种算法好.

表 3-4 ES、GA、IGA、HIA 对 f_2 30 次运行结果比较

函数 $f_2, \varepsilon_1 = 79, \varepsilon_2 = 78, n = 5$						
	NF	NS	BV	WV	AV	AE
ES	22	8	79.00	75.00	76.80	5750
GA	30	0	73.00	67.00	69.433	12120
IGA	0	30	80.00	79.00	79.367	17897
IHIA	0	30	80	79.00	79.60	24193

函数 f_3 为有名的 Rosenbrock 测试函数, 从表 3-5 中所列数据可看出, ES 比 GA、IGA、IHIA 好, IHIA 比 GA、IGA 好; 从平均计算次数看, IHIA 的计算次数最低, 此情形下, IHIA 比 ES 稍次, 但优于 GA 及 IGA.

表 3-5 ES、GA、IGA、HIA 对 f_3 30 次运行结果比较

函数 $f_3, \varepsilon_1 = 9.85, \varepsilon_2 = 9.8, n = 5$						
	NF	NS	BV	WV	AV	AE
ES	0	30	9.985	9.974	9.981	51114
GA	30	0	8.219	5.930	7.886	12300
IGA	26	4	9.992	7.359	8.999	20600
IHIA	0	30	9.8828	9.81267	9.87216	6482

表 3-6 ES、GA、IGA、HIA 对 f_4 的 30 次运行结果比较

函数 $f_4, \varepsilon_1 = 0.999, \varepsilon_2 = 0.999, n = 5$						
	NF	NS	BV	WV	AV	AE
ES	19	11	1.00	0.128	0.448	5750
GA	30	0	0.184	0.120	0.129	12164
IGA	0	30	1.00	0.997	1.00	22792
IHIA	0	30	1	1	1	14333

函数 $f_4, \varepsilon_1 = 0.999, \varepsilon_2 = 0.999, n = 7$						
	NF	NS	BV	WV	AV	AE
ES	30	0	0.128	0.128	0.128	7990
GA	30	0	0.128	0.071	0.094	12124
IGA	16	14	1.00	0.128	0.563	23009
IHIA	0	30	1	1	1	5866

f_4 为多峰值函数, 表 3-6 表明 IHIA 对于 f_4 的 5 维及 7 维 30 次实验均获得最优解, 且 IHIA 比 IGA 的平均计算次数明显低, 因此 IHIA 所获最好解比 GA、IGA 的最好结果稍次, 但 IHIA 所获最差结果及平均值均明显比其余三种算法的结果好。由 IHIA 对 f_4 的测试获知, IHIA 比其余三算法处理多峰值优化问题更具有优越性。

表 3-7 ES、GA、IGA、HIA 对 f_5 的 30 次运行结果比较

函数 $f_5, \varepsilon_1 = 950, \varepsilon_2 = 949.5, n = 5$						
	NF	NS	BV	WV	AV	AE
ES	30	0	947.02	927.12	937.30	5750
GA	21	9	949.98	947.87	949.10	15728
IGA	0	30	949.99	949.77	949.85	22638
IHIA	0	30	950	950	950	20673

函数 $f_5, \varepsilon_1 = 995, \varepsilon_2 = 995, n = 10$						
	NF	NS	BV	WV	AV	AE
ES	30	0	987.07	941.30	967.63	11500
GA	9	21	999.95	985.56	995.31	38900
IGA	4	26	999.92	994.85	996.67	26349
IHIA	0	30	997.668	995.1	997.23	28237

函数 f_5 为多峰值函数, 表 3-7 说明, IHIA 对于 7 维的 f_5 进行 30 次实验均

获得最优解,且 IHIA 比 IGA 的平均计算次数明显低;对于 10 维的 f_5 ,由于要求 IHIA 在获得满足要求精度后算法运行结束,因此 IHIA 所获最好解比 GA、IGA 的最好结果稍次,但 IHIA 所获最差结果及平均值均明显比其余三种算法的结果好。由 IHIA 对 f_5 的测试获知, IHIA 比其余三算法处理多峰值优化问题更具有优越性。

从以上的对比分析可看出, IHIA 对所有测试问题 30 次实验过程中,其失败次数最低,成功率最高;整体上 IHIA 所获最好值、最差值、平均值均比其余三种算法的结果好,平均计算次数明显比 IGA 的低。由此表明 IHIA 在处理高维多峰函数优化方面具有明显优势。

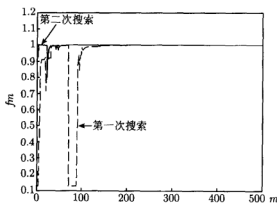


图 3-24 IHIA 对 5 维函数 f_4 连续两次搜索最大值的进程比较

另外,从免疫应答中体液免疫应答原理获知,初次免疫应答时,人体的抗体数量及抗体应答能力皆弱,但再次免疫应答时,由于记忆细胞的作用,抗体应答明显增强,抗原被清除较快。这种理论已在本文免疫算法中得到刻画。在此算法中,记忆池对算法处理相同或相似的优化问题能明显提高搜索最优解的速度。以 IHIA 对 5 维函数 f_4 为例说明记忆池对算法搜索性能的作用,所获结果如图 3-24 所示。

第十四节 应用举例

例 3.14.1 非齐次免疫算法在 TSP 中的应用

TSP 的数学描述可参见例 2.5.1。这个问题的特点是: (1) 目标函数为非连续的离散函数; (2) 函数所依赖的定义区域由比较分散的点构成,并且所有这样的点构成的空间较大,即含 $n!$ 个元素,从数学规划的角度极难探讨最优解的存在问题及寻找最优解。以下以著名的 30 城市及 75 城市为例说明算法 IHIA 的应用。该算法的参数选定: 群体规模为 100, 浓度抑制半径 σ 为 0.15, 初始温度 T_0 为 200, 克隆选择率 α 为 0.1, 细胞繁殖数目 M 为 100, 随机产生新抗体率 μ 为 0.08。突变规则选为换位算子和逆转算子。该算法获城市的实际路径如图 3-25 所示。

从搜索效果看,对 30 城市获最优路径长度为 423.741,对 75 城市获最佳路径长度为 545.366。尽管 TSP 为多字符的组合优化问题,免疫算法也能获得比较满意的结果。

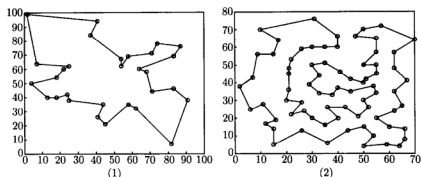


图 3-25 (1) 30 城市的实际路径; (2) 75 城市的实际路径

例 3.14.2 系统辨识问题

状态空间模型形式如下^[19]:

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} \theta_1 x_1(t) x_2(t) \\ \theta_2 x_1^2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ u(t) \end{bmatrix},$$

$$y(t) = \theta_3 x_2(t) - \theta_4 x_1^2(t),$$

$$x_1(0) = 1, x_2(0) = 1, t = 0, 1, \dots, 50.$$

实验结果及比较如表 3-8 所示。

由以上优化结果可知, 免疫算法对于解决函数优化及组合优化问题具有一定的有效性, 该算法具有开采与探测的特点, 同时体现了体液免疫应答中克隆选择原理、亲和成熟机理及独特型免疫网络原理的特征。但对于函数优化问题, 由于此类问题的变量的取值具有连续性, 因此可采用实数编码改进免疫算法, 以便提高算法搜索最优解的速度, 这将在第四章提出几种类型的形态空间上免疫算法。

表 3-8 算法估计值比较

参数	θ_1	θ_2	θ_3	θ_4
真实值	0.5	0.3	1.8	0.9
文献 [19] 估计值	0.4916	0.30141	1.8432	0.9267
IHIA 估计值	0.500489	0.301075	1.79863	0.899316

第十五节 本章小结

提取体液免疫应答中免疫机理, 将各机理设计为免疫算子, 并根据体液免疫应答作用过程构建运行机制框架, 进而获得人工免疫系统中最为重要的研究分支的一般框架: 一般性免疫算法。给予其严格定义, 探讨其收敛性、收敛速度估计、稳定性、鲁棒性及计算复杂度, 获得其系统性的理论结果, 实验论证了理论结果的

正确性。针对免疫选择算子的选择不同, 对齐次及非齐次免疫算法进行性能测试。应用高维优化问题, 将非齐次免疫算法与遗传算法、进化策略及免疫遗传算法进行比较, 获得该算法的优越性。最后将非齐次免疫算法应用于 30、75 城市的 TSP 及系统辨识, 获得了满意结果。

参 考 文 献

- [1] 张文修, 梁怡. 遗传算法的数学基础. 西安: 西安交通大学出版社, 2001. 214
- [2] Rudolph G. On a Multiobjective Evolutionary Algorithm and Its Convergence to the Pareto Set. Proc. of the 1998 IEEE International Conference on Evolutionary Computation, Piscataway (NJ), 1998, 511~516
- [3] Rudolph G. Convergence Rates of Evolutionary Algorithms for a Class of Convex Objective Functions. Control and Cybernetics, 1997, 26(3): 375~390
- [4] Rudolph G. Convergence Properties of Some Multiobjective Evolutionary Algorithms. In Proc. of the 2000 Conference on Evolutionary Computation, Piscataway, New Jersey, July 2000, 2: 1010~1016
- [5] Rudolph G. Finite Markov Chain Results in Evolutionary Computation: A Tour Horizon. Fundamenticae 1998, 35(1-4): 67~89
- [6] Rudolph G. Some Theoretical Properties of Evolutionary Algorithms under Partially Ordered Fitness Values. Proc. of the Evolutionary Algorithms Workshop (EAW-2001), Bucharest, Jan. 2001
- [7] Rudolph G. A Partial Order Approach to Noisy Fitness Functions. In Congress on Evolutionary Computation, 2001, 318~325
- [8] Kursawe F. A variant of evolution strategies for vector optimization. In H. P. Schwefel and Manner, editors. Parallel Problem Solving from Nature. Springer, Berlin and Heidelberg, 1991, 193~197
- [9] Sibylle D. Müller S D, Walther J H, et al. Evolution Strategies for the Optimization of Microdevices. Proc. of Evolutionary Computation (CEC 2001), 2001, 302~309
- [10] Müller S D, Schraudolph N N, Koumoutsakos P D. Step Size Adaptation in Evolution Strategies using Reinforcement Learning. Proc. of Evolutionary Computation (WCCI'02), Honolulu, 55~58
- [11] 王凌. 智能优化算法及其应用. 北京: 清华大学出版社, 施普林格出版社, 2001. 230
- [12] Jiao Licheng, Wang Lei. A Novel Genetic algorithm Based on Immunity. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2000, 30(5): 552~561
- [13] 陈国良, 王熙法, 庄镇泉等. 遗传算法及其应用. 北京: 人民邮电出版社, 2001. 433
- [14] 周明, 孙树栋. 遗传算法及应用. 北京: 国防工业出版社, 2000. 202
- [15] 谭志扬. 人工免疫算法在 Flow-shop 问题中的应用. 计算机工程与应用, 2002.14, 98~99
- [16] de Castro L N, Von Zuben F J. Artificial Immune Systems: Part I—Basic Theory and Applications. Technical Report, TR-DCA 01/99, 1999. 95
- [17] Chung, J S, Jung H K, Hahn, S Y. A Study on Comparison of Optimization Performances between immune Algorithm and other Algorithms. IEEE Transactions on magnetics, 1998, 34(5): 2972~2975
- [18] 逊崔学. 基于免疫原理的多目标进化算法群体多样性研究. 模式识别与人工智能, 2001, 14(3): 291~295
- [19] 姜波, 汪秉文. 基于遗传算法的非线性系统工程模型参数估计. 控制理论与应用, 2000, 17(1): 150~152

第四章 形态空间上免疫算法及收敛性理论

第一节 引言

在第三章,基于体液免疫应答过程,建立了免疫算法的一般框架,并通过测试函数及 TSP 实验获得此算法的有效性,但该算法是建立在字符串编码基础上的,在用于函数优化方面,存在着与其他基于字符串编码的智能算法所具有的同样不足是,算法存在编码和解码问题,以及字符串所描述的解的基因(即字符)的突变有可能导致该解突变性很大,从而使算法搜索速度受到影响.对于第二章描述的基于连续变量的函数优化问题(NFP)及(CFP),可根据变量取值的连续性特性,构建基于实数编码的免疫算法解决此类问题,以便提高算法搜索最优解的速度.

目前,在基于免疫学原理及实数编码的免疫算法研究方面,文献[1]利用克隆选择原理建立了基于实数编码的广义克隆选择算法,该算法的关键在于引入自适应突变算子,算法的设计类似于进化策略中的 (μ, λ) 策略,突出的优点是充分利用并行搜索及邻域搜索的特点提高算法的搜索性能.其突出不足是,突变的克隆群中仅有亲和力和有改进的克隆取代其母体,其他克隆被清除.这种思想导致突变的克隆群中对群体进化过程有益的克隆被清除,进而使算法搜索性能受到影响.文献[2]从免疫网络的角度,借助文献[3]的克隆选择算法及免疫网络中抑制思想,设计了一种用于函数优化的免疫算法,此算法的关键在于亲和突变、克隆群评价及克隆抑制设计,算法的突出优点是:(1)合理地体现了克隆选择原理及网络确定性抑制的有效特征;(2)算法采用并行的思想从多方位搜索整体最优解;(3)利用平均适应度改进的方法对克隆群重复突变,增强局部搜索能力;(4)算法搜索过程中群体规模被动态调整.该算法存在的问题是:(1)计算复杂度较高,即为 $O((NN_c)^2)$,其中 N 为群体规模, N_c 为每一个抗体的克隆数目;(2)算法的结构复杂,不便于实际应用;(3)算法的设计仅考虑了应用方面,而未顾及理论探讨.由此可见,基于实数编码,设计具有应用有效、结构简单、易于嵌入于其他智能算法等特点的免疫算法有待进一步探讨.其次,文献[4~8]利用特定的约束处理方法,并借助遗传算法或免疫进化机制解决了单目标约束优化问题.特别文献[8]将进化群体划分为抗体群和抗原群,并设计特定的抗体适应度计算方案,以及利用遗传算法进化抗体群,并最终获得满足约束的最优解,这种方法提供了探讨用免疫算法解决约束优化问题的新思路.

综上所述获知,基于免疫机理的算法解决约束优化问题的研究仍处于起步阶段,深入挖掘免疫系统的内在机制设计有效地解决这类优化问题的算法的研究有

待深入。

本章提出几种不同类型的形态空间上免疫算法, 这些算法在一定程度上是对第三章建立的免疫算法的变形, 但算子的设计有所不同。在以下构建的算法与优化问题的对应中, B 细胞视为抗体, T 细胞视为算法设计中的参数, 并在第二节至第四节的算法中, 假定 T 细胞已被确定。抗体视为优化问题的可行解, 抗原视为优化问题本身或进化群体中最好的解, 这依据所构建的算法而论。

由于任何闭区间 $[a, b]$ 均可经过可逆变换变为区间 $[0, 1]$, 因此不妨设问题 (NFP) 及 (CFP) 中可行解的变量均在 $[0, 1]$ 上取值。在以下的算法构建中, 抗体为由优化问题 (NFP) 或 (CFP) 的可行解中的 p 个变量按一定的次序排列所构成的序列, 即 (x_1, x_2, \dots, x_p) , 每一变量 x_i 称为染色体, 每一染色体由 0 到 9 之间的整数作为染色体的基因构成。抗体对抗原的亲合力及抗体的激励度的设计视以下各算法而定。值得一提的是, 第三章所定义的免疫算子, 若在本章的算法中被引用, 其定义与前相同, 但算子的具体设计有所区别。

第二节 小生境免疫算法

小生境免疫算法是将进化论中小生境概念与体液免疫应答的部分机理相结合而提出的一种新智能算法, 并用于处理优化问题 (NFP)。小生境意指生物在特定环境下的生存环境^[9]。这个概念的主要思想是“人以群分, 物以类聚”, 即反映了在大自然的进化过程中, 各种生物在特定环境下生存, 同种生物中存在着优秀的生物, 各生物之间存在着相互竞争, 不同种生物之间又存在着信息交换, “资源共享”体现了在特定环境中共同生存的同种生物分享有限的资源, 这些生物之间通过相互协调达到共同进化, 各生物之间也存在着相互抑制的机制, 对于适应环境能力弱的生物, 在资源不足的前提下, 将会逐渐被淘汰。受到这些思想的启发, Golerge 于 1989 年提出一种共享适应度小生境实现方法 (简称为共享机制), 该方法用于智能算法的构建中可增强群体多样性。

在小生境免疫算法中, 抗原被视为问题 (NFP), 抗体对应此问题的候选解, 抗原对抗体 Ab 的亲合力 $aff(Ab)$ 与定义 3.3.1 相同。群体规模指定为 N 。免疫算子包括克隆选择、亲和突变、募集新成员; 另外基于“人以群分, 物以类聚, 资源共享”的思想, 利用共享适应度小生境实现方法^[9]中共享函数, 构建具体获取多种记忆细胞的共享机制算法, 其目的是增强群体多样性及保存优良个体, 提高算法搜索性能。利用免疫算子及共享机制算法可构建小生境免疫算法 (NIA: niched immune algorithm)。该算法的主要思想可简要描述为: 首先记忆当前抗体群, 并利用共享机制算法获记忆细胞群; 然后选出当前抗体群中较高亲和力的抗体进行突变; 最后将记忆细胞与已突变的抗体组合在一起, 并随机产生新抗体更新低亲和力的抗体, 获新抗体群。为便于叙述小生境免疫算法, 在此首先引入共享机制算法 (针对问题

(NFP)).

一、共享机制算法

Step1 设抗体群为 $P=\{Ab_1, Ab_2, \dots, Ab_N\}$, 计算抗体适应度 (即亲和力) $aff(Ab)$, $Ab \in P$.

Step 2 按下列步骤确定小生境及优良抗体:

Step 2.1 置 $i = 1$;

Step 2.2 计算抗体间的距离

$$d_{ik} = ||Ab_i - Ab_k||, k = i, i+1, \dots, N;$$

Step 2.3 根据 $d_{ik} < \sigma_0$, $k = i, i+1, \dots, N$, 确定小生境子群 M_{p_i} , p_i 为 M_{p_i} 的元素个数, 其中 σ_0 依据问题 (NFP) 而定;

Step 2.4 若 M_{p_i} 中包含 P 中适应度最高的抗体, 则保存其中一个最高适应度的抗体, 其余抗体按 Step2.5 更新其适应度; 否则, 按 Step2.5 更新 M_{p_i} 中抗体的适应度;

Step 2.5 计算 M_{p_i} 中抗体的更新适应度 (即共享适应度)

$$f'_j = f_j / \sum_{s=i}^{i+p_i-1} shd_{js}, j = i, i+1, \dots, i+p_i-1,$$

其中

$$Shd_{js} = \begin{cases} 1, & d_{js} = 0, \\ 1 - (d_{js}/\sigma)^\lambda, & d_{js} < \sigma, \\ 0, & \text{否则;} \end{cases}$$

Step 2.6 利用更新适应度 f'_j 及处罚函数 Penalty 对该子群中低适应度的抗体进行处罚, 即当 $Ab_j, Ab_k \in M_{p_i}$, $||Ab_j - Ab_k|| < L$, $L < \sigma$ 时, 比较两个抗体的适应度, 并对其中适应度较低的抗体进行处罚,

$$f_{\min}(Ab_j, Ab_k) = \text{Penalty}, \quad j, k = i, i+1, \dots, i+p_i-1;$$

Step 2.7 当 $i + p_i < N$ 时, 置 $i \leftarrow i + p_i$, 返回步骤 2.2; 否则, 进入下一步.

Step 3 确定记忆抗体集, 按从大到小的次序排列各抗体的适应度, 选择前 M 个抗体构成记忆抗体集 P_M .

以上算法通过利用共享适应度调整抗体的适应度, 抑制浓度较高的抗体, 但其中适应度最高的一个抗体不受抑制的影响. 设计此操作的目的在于增强群体的多样性, 同时保存群体中最好的抗体. 算法中的参数 λ, σ 视具体优化问题而定, 于是利用此算法及几种免疫机制可构建处理问题 (NFP) 的算法如下.

小生境免疫算法

Step 1 确定进化代数 $n \leftarrow 1$, 随机产生 N 个初始抗体构成初始群体 A_n .

Step 2 记忆细胞获取. 将共享机制算法作用于 A_n , 获 M 个记忆细胞构成群体 M_n .

Step 3 克隆选择. 按选择率 σ , 选择 N_0 个较高亲和力的抗体构成抗体群 B_n , 其中 $N_0 \equiv \text{round}(\sigma \cdot N)$, σ 满足 $\text{round}(\sigma \cdot N) + M = N$.

Step 4 亲和突变. B_n 中抗体 Ab_i 被突变的概率为 α_i , 在突变的情形下, 抗体的各个染色体 (即可行解的分量) 分别进行均匀突变, 从而获抗体群 C_n , 其中

$$\alpha_i = 1 - \exp\left(-\frac{\mu_n + \max_n - \text{aff}(Ab_i)}{\mu_n + \max_n - \min_n}\right), \mu_n = \mu_0 + \frac{T_0}{n},$$

$$\max_n(\min_n) = \max(\min)_{Ab \in B_n}(\text{aff}(Ab)),$$

这里 μ_0, T_0 为可调节参数.

Step 5 抗体组合及更新. 随机产生 d 个新抗体取代 $C_n \cup M_n$ 中 d 个低亲和力抗体, 获新抗体群 A_{n+1} , 其中 $d \equiv \text{round}(\mu \cdot N)$.

Step 6 终止条件判断. 若不满足终止条件, 则继续返回步骤 2; 否则, 输出结果.

评注 4.2.1 此算法中并未引入细胞克隆操作, 目的在于提高寻优速度. Step 4 在于提高抗体的亲和力, 其主要基于抗体的突变率与其亲和力成反比的机理设计; Step 5 产生 d 个新抗体主要是防止算法陷于局部搜索, μ 一般应在 0.05 至 0.08 之间.

算法 NIA 与基本遗传算法 SGA 的主要区别如下:

(1) SGA 通过交叉算子增强群体多样性, 并使群体中个体突变的可能性较小, 而 NIA 通过记忆细胞、亲和突变及新抗体的引入实现群体多样性, 抗体的突变受其亲和力制约, 低亲和力的抗体突变的概率较大;

(2) SGA 需设计特定的适应度函数及按概率随机选择, 而 NIA 可直接作用于目标函数且按确定性方式选择抗体参与进化;

(3) SGA 对初始群体的分布有敏感性, 而 NIA 却无敏感性;

(4) SGA 是随机搜索算法, NIA 是确定性与随机性相结合的搜索算法;

(5) GA 是由单一途径获新一代群体, 而 NIA 是多途径获新一代群体, 即由记忆细胞、进化所获的抗体及随机产生的新抗体构成;

(6) GA 是自封闭式进化, 而 NIA 是开放式进化, 即随时有自我抗体参与.

二、小生境免疫算法的性能测试

为了验证算法 NIA 的有效性, 将其与克隆选择算法 (CLONALG)^[3] 及遗传算

法 (REGA)^[10] 进行比较。此三种算法用于例 2.5.2 中的测试函数 ($F_1 \sim F_4$) 及下列非连续函数 (即 De Jong 函数) 进行测试:

$$F_5: f(x_1, x_2, \dots, x_5) = \sum_{i=1}^5 \text{int } \text{eger}(x_i), -5.12 \leq x_i \leq 5.12, i = 1, 2, 3, 4, 5$$

测试内容包括: (1) 求函数 ($F_1 \sim F_4$) 的最大值、 F_5 的最小值; (2) 算法对测试函数搜索最优解的收敛速度。

为了公平比较, 指定此三种算法的群体规模为 80, 迭代次数为 100。其余各参数选定为使各算法获最佳效果时的参数, 即 REGA 的交叉概率为 0.6, 突变概率为 0.001, CLONALG 的每一代随机产生的新抗体数为 6; NIA 的参数选定为: $M=20$, $\sigma=0.8$, $\mu=0.6$ 。 $F_1 \sim F_4$ 的最大值可参见例 2.5.2, F_5 的最小值为 -30。将上面的三种算法用于上所叙述的 5 个函数求最值, 并独立运行 10 次, 获知 NIA 对 F_4 及 F_5 每一次皆获多个最优解 (因这两函数包含多个最优解), 而 CLONALG 及 REGA 却较困难。另外, 用下式刻划各算法的群体多样性, 即对同一问题, 群体多样性的平均变化率为

$$P = \frac{10 \text{ 次独立运行到指定的迭代数时不同的抗体数之和}}{10 \times N},$$

于是可获表 4-1。

表 4-1 三算法用于上五函数优化, 所获最优解或次最优解及群体多样性比较

函 数	算法 aiA			算法 CLONALG			算法 REGA		
	最好解	函数值	P 值	最好解	函数值	P 值	最好解	函数值	P 值
F_1	(-2.048, 2.048)	3905.93	0.935	(-2.048, -2.048)	3905.93	0.18	(-2.03599, -2.044)	3839.89	0.89385
F_2	(-4, -2)	0.926164	0.94175	(-4.10577, -2.63937)	0.95022	0.0625	(-6.74487, 4.00782)	0.943635	0.8625
F_3	(-0.8, -0.8)	210.482	0.837	(-7, -0.8)	193.602	0.462	(5.38612, 5.54252)	179.606	0.525
F_4	(-3, -2), (3, 2)	162.9	0.93625	(-3, -2)	162.9	0.29375	(-3, -1.99218)	162.006	0.89125
F_5	每次至少 12 个最优解	-30	0.92125	每次至少 2 个最优解	-30	0.91125	每次最多 2 个最好解	-28, -29 或 -30	0.9625

由表 4-1 获知, 在各算法的终止代数 100 的条件下, NIA 除了对函数 F_2 仅能获次优解外, 对其他 4 个函数皆获最优解。特别地, NIA 能搜索到函数 F_4 、 F_5 的多个最优解。从搜索效果看, NIA 优于 CLONALG, CLONALG 优于 REGA。其次, NIA 比 CLONALG 的群体多样性好, 且整体上也优于 REGA 的多样性。由此分析获知, NIA 在处理多模态函数及非连续的优化问题方面优于 REGA 和

CLONALG, 这也说明免疫算法比遗传算法有更多的优越性。另外, 为了说明这三种算法搜索最优解的速度差异, 选取测试函数 $F_2 \sim F_5$, 将此三种算法分别作用于测试函数并分别独立运行 10 次, 且分别随机选择一次运行结果进行比较获图 4-1。在此图中, f_n 表示算法搜索过程中, 第 n 代群体的最大目标函数值, 其中对于图 4-1(4), f_n 表示最小目标函数值。

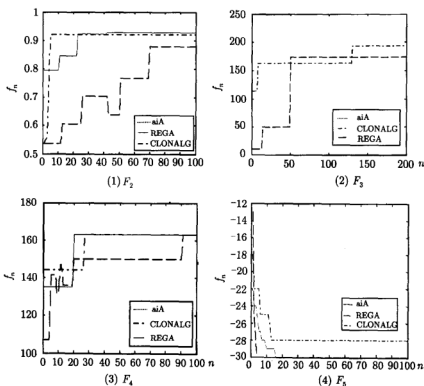


图 4-1 NIA、CLONALG、REGA 分别用于 $F_2 \sim F_4$ 搜索最大值及 F_5 的最小值的进程

由图 4-1 可知, 对于 $F_3 \sim F_5$, NIA 比 CLONALG 较快地搜索到最优解; 对于 F_2 , CLONALG 比 NIA 搜索到最优解的速度快。一般来说, NIA 及 CLONALG 在迭代次数为 100 代范围内可获最优解或次最优解, 而 REGA 在迭代次数为 100 范围内至多获次优解 (对于 F_5 , 选择 REGA 收敛的情形进行比较), 这说明小生境免疫算法是行之有效的智能计算方法, 其优点在于结构简单且不仅能快速搜索到最优解, 而且最优解一旦被搜索到, 其将继续保持, 同时能搜索到多个最优解 (对于多最优解测试问题)。

第三节 动态规模免疫算法

本节建立一种有别于小生境免疫算法的形态空间上免疫算法, 即动态规模免

算算法 (VIA: variable immune algorithm). 该算法是依据体液免疫应答的部分机理提出, 算法的关键是设计基于实数编码的突变规则, 以及引入抗体浓度机制设计亲和力和评价方案, 其突出特点是: (1) 抗体浓度机制被引入用于设计抗体亲和力; (2) 进化群体的规模自适应地被调整.

一、动态规模免疫算法描述

在算法设计中, 抗体对应优化问题 (NFP) 的可行解, 抗原对应进化抗体群中最好的解. 抗体及抗原的编码与第一节的描述相同. 设 Ag 表示抗原, X 表示含 N 个抗体构成的集合, 即 $X = \{Ab_i, 1 \leq i \leq N\}$, $m_i(\delta)$ 表示 Ab_i 的 δ 邻域内含 X 中抗体的个数, 则抗体 Ab_i 的浓度定义为 $C_i = \frac{m_i(\delta)}{N}$, 于是 Ab_i 与 Ag 的亲合力定义为

$$F(Ab_i) = \frac{1}{1 + \|Ab_i - Ag\|} \exp(-kC_i), i = 1, 2, \dots, N, \quad (4.3.1)$$

其中 k 为调节因子. 式 (4.3.1) 表明, 对与抗原有较好匹配, 但浓度高的抗体将会受到抑制; 对既与抗原有较好匹配, 又有较低浓度的抗体将会获得鼓励; 对既与抗原较差匹配, 且浓度较高的抗体将会受到排斥. 此设计的目的在于维持群体多样性, 于是动态免疫算法可描述如下:

Step 1 确定进化代数 $n \leftarrow 0$, 随机产生规模为 N_0 的初始群体 A_0 .

Step 2 复制 A_n 中最好的一个抗体作为抗原 Ag_n .

Step 3 选择 A_n 中 $\text{round}(\alpha|A_n|)$ 个较好抗体构成群体 A_{n1} , 其余部分构成 A_{n2} .

Step 4 对 A_{n1} 中每一抗体 Ab 按突变率 α 突变, 即: 设抗体 Ab 表示为 $Ab = (x_1, x_2, \dots, x_p)$, 抗原 Ag_n 表示为 $Ag_n = (y_{n1}, y_{n2}, \dots, y_{np})$, Ab 按下式进行超突变:

$$Ab \leftarrow Ab + \beta(Ag - Ab), \beta \in [0, \alpha], \alpha = 1 - \exp(-\|Ab_i - Ag_n\|),$$

其中 β 为 $[0, \alpha]$ 上的随机数, 从而获 A_{n11} .

Step 5 根据

$$\|f(Ab_i) - f(Ab_j)\| < \sigma,$$

将 A_{n2} 划分为互不相交的子群, 对各子群中较差抗体进行处罚, 所有子群中未被处罚的抗体构成群体 A_{n21} , 其中 σ 依据群体规模 $|A_n|$ 自适应调整.

Step 6 随机产生 $d_n \equiv \text{round}(\mu|A_n|)$ 个新抗体插入 $A_{n11} \cup A_{n21}$, 获抗体群 A_{n+1} .

Step 7 若满足终止条件, 则最终的抗原即为最优解; 否则, 则返回 Step2.

由以上算法描述获知, 算法 VIA 与 NIA 的主要区别在于: (1) DIA 中抗体的亲和力刻画了抗体的浓度; (2) DIA 通过对未被选择突变的抗体构成的群体引入网

络抑制保存多种不同的抗体, 而 NIA 通过共享机制算法保存多种抗体; (3) DIA 的群体规模动态调整, 而 NIA 的群体规模固定不变。

二、动态规模免疫算法性能测试

为了测试算法 VIA 解决函数优化问题的有效性, 将其与算法 REGA^[10] 用于例 2.5.2 中的函数 F_1 、 F_4 及下列函数^[11] 求最大值比较:

$$G_3: \quad f(x, y) = x \sin(4\pi x) - y \sin(4\pi y + \pi) + 1, \quad -1 \leq x, y \leq 2.$$

函数 G_3 是典型的多模态函数, 该函数的最大值为 4.25389, 最优解为 (1.6288, 1.62889)。该函数有许多取局部极大值的解, 特别在取最大值的最优解附近包含有较多的局部最优解, 因此搜索此函数的最优解比较困难。VIA 的参数选定范围为

$$100 \leq N_0 \leq 200, \quad 0.5 \leq \alpha \leq 0.7, \quad 0.1 \leq \sigma \leq 0.5, \quad 0.05 \leq \mu \leq 0.08.$$

在此次实验中, VIA 的参数选为 $N_0 = 120$, $\alpha = 0.52$, $\sigma = 0.2$, $\mu = 0.08$ 。算法 REGA 的参数选择是, 群体规模固定为 80, 交叉概率为 0.6, 突变概率为 0.001。两算法的终止迭代数均为 120。测试的内容包含收敛速度及这两个算法对测试问题分别独立运行 30 次获最优解及次优解的比较。尽管测试函数为低维空间上的多模态函数, 但可给予用算法 VIA 解决高维空间优化问题的启示。为便于区别, 用“*”及“0”分别表示 VIA 和 REGA 所获的最优解或次优解。于是 VIA 和 REGA 分别对同一测试问题独立连续运行 30 次的结果比较如图 4-2 所示, 图中 n 表示算法独立运行的次数, $\text{opt}(n)$ 表示第 n 次所获的最优解或次优解的函数值。

由图 4-2 获知, DIA 对 F_1 、 F_4 、 G_3 几乎每次皆可获最优解, 而 REGA 却获最优解的次数较少, 它仅在迭代数较大时才能获最优解。总体上, VIA 获最优解的次数明显高于 REGA 所获最优解的次数。

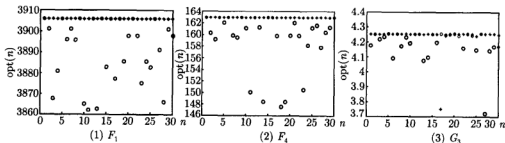


图 4-2 算法 VIA 及 REGA 分别用于函数 F_1 、 F_4 、 G_3 获最优解及次优解比较

另外, 为了反映 VIA 搜索最优解的速度, 将 VIA 与 REGA 对 F_1 、 F_4 、 G_3 搜索最优解的速度进行比较。对于每一测试函数, 在 VIA 和 REGA 分别独立运行

的 30 次中, 随机抽取一次运行结果及抽取 REGA 搜索到最优解的一次结果进行比较, 可获图 4-3.

由图 4-3 知, 算法 DIA 对每一测试函数均比算法 REGA 搜索到最优解的速度快, 且约在第 50 代左右就能获最优解, 但 REGA 搜索到最优解的迭代数较大. 这表明基于实数编码的免疫算法 DIA 在局部区域搜索及最优解的搜索速度方面, 极大优于最优保存的遗传算法, 但该算法的参数 α , σ , μ 对算法的搜索性能较重要. 至于如何合理地体现此三个参数的关系, 将在第五节通过建立模糊控制器动态调节各参数.

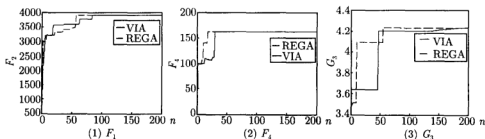


图 4-3 VIA 及 REGA 分别用于函数 F_1 、 F_4 、 G_3 搜索最优解的速度比较

第四节 约束优化免疫算法

非线性约束优化问题 (第二章第五节的问题 (CFP)) 是智能优化算法研究的主要问题, 解决这类问题的方法已较多, 其中大部分方法是基于遗传算法而提出的 [4~6]. 在这些方法中, 各自有独特的约束条件处理方法, 特别文献 [6] 提出一种模糊处罚函数法自适应处罚非法解的呈现, 这种方法是对处罚函数法的改进. 另外关于利用免疫系统的特征与遗传算法相结合, 提出免疫遗传算法处理这类问题的研究已初步涉及, 如文献 [7, 8] 将进化群体划分为抗体群和抗原群, 并通过设计适应度计算方案利用遗传算法对抗体群进行进化, 最终获得的抗体群中最好的抗体则为最优解, 这种方法提供了利用免疫学原理构建免疫算法解决问题 (CFP) 的一些启迪. 在此部分, 利用免疫系统中抗体浓度概念设计抗体亲和力, 借助数值计算中黄金分割法设计特定约束处理算子, 利用区域等分思想设计产生满足约束条件的有效解集, 进而提取免疫应答的几种进化机制并将其组合获约束优化免疫算法 (COIA: constrained optimization immune algorithm) 解决问题 (CFP).

一、问题描述及转化

一般非线性约束函数优化问题的极小化模型可描述为第二章第四节中的问题 (CFP). 以下考虑这类问题的如下形式:

$$\min_{x \in \tilde{D}} f(x),$$

$$(CFP) \quad \tilde{D} = \left\{ \begin{array}{l} x = (x_1, x_2, \dots, x_p) \in D | h_i(x) = 0, \\ g_j(x) \geq 0, \quad 1 \leq i \leq L, \quad 1 \leq j \leq M, \quad a_k \leq x_k \leq b_k, \quad 1 \leq k \leq p, \end{array} \right.$$

其中 $D \subset R^p$, \tilde{D} 称为约束集, $x \in \tilde{D}$ 称为有效解, $x \in D \setminus \tilde{D}$ 为非法解, 不妨设 $f(x) \geq 0$. 对于问题 (CFP), 通常在算法搜索中要求进化解为有效解极为困难, 甚至属于不可能. 为了回避这种问题, 通常采用处罚函数法处理约束条件. 若应用静态处罚法, 则出现处罚因子的确定较为困难, 若其过小, 则易于获局部最优解, 反之则导致计算速度慢, 这结论可由下例获知.

$$\min_{(x,y) \in R^2} (x^2 + y^2),$$

$$x + 2y = 1, \quad 3x + y \leq 2, \quad x \geq 0,$$

该问题的最小值为 $\frac{1}{5}$, 最优解为 $\left(\frac{1}{5}, \frac{2}{5}\right)$. 若罚函数取为

$$F(x, y) = x^2 + y^2 + \lambda(x + 2y - 1)^2, \lambda > 0,$$

由极值的必要条件获得在约束区域

$$\{(x, y) \in R^2 | 3x + y \leq 2\}$$

上取最小值的解为 $\left(\frac{\lambda}{1+5\lambda}, \frac{2\lambda}{1+5\lambda}\right)$, 最小值为 $g(\lambda) = \frac{5\lambda^2 + \lambda}{(1+5\lambda)^2}$. 由此式获知 $\lim_{\lambda \rightarrow \infty} g(\lambda) = \frac{1}{5}$, 这表明处罚因子的选择对求解最优解具有重要影响. 为此采用动态处罚法处理问题 (CFP) 中等式约束, 问题 (CFP) 变为

$$\min_{x \in \tilde{D}} F(x) = f(x) + \sum_{i=1}^L \lambda_i(x) h_i^2(x),$$

$$(P) \quad \tilde{D} = \{x \in D | g_j(x) \leq 0, 1 \leq j \leq M\},$$

$$\lambda_i(x) = \begin{cases} |h_i(x)|, & |h_i(x)| \leq \varepsilon, \\ p|h(x)|, & \varepsilon < |h_i(x)| < M_i, \\ ph^2(x), & |h_i(x)| \geq M_i, \end{cases}$$

在此选择 $\varepsilon = 0.01$, $p = 1500$, $M_i = 10$. 通过以上的转换, 一般的优化问题 (CFP) 可转化为仅含不等式约束的极小化问题, 不妨设为

$$(Q) \quad \min_{x \in X} f(x),$$

$$X = \{x \in D | g_j(x) \geq 0, 1 \leq j \leq M\}.$$

二、约束优化免疫算法原理

将体液免疫应答的部分机制设计为免疫算子, 对非均匀突变^[12]作适当改进以及引入邻域搜索思想设计特定亲和突变算子, 引入最优化理论中黄金分割法设计个体重构算子, 其起到提高寻优效率及在线修正非法解的作用, 进而将这些算子有机组合, 便可获约束优化免疫算法. 该算法由五种免疫算子构成, 即免疫选择、亲和突变、记忆细胞演化、免疫检测及募集新成员, 其中免疫检测由有效检测和个体重构组成. 亲和突变使抗体有指导地进行突变, 抑制盲目突变产生的负面影响, 有效检测为在当前群体的抗体与其突变所获抗体中选择亲和力最高的抗体作为子代抗体, 个体重构使自反应细胞变为抗体, 目的在于防止群体出现退化现象, 及时修正无效个体, 个体重构属于特定的非法解修正方法. 该算法的流程图如图 4-4 所示.

在算法设计中, 抗原被视为问题 (Q), 抗体被视为有效解, 自反应细胞对应非法解. 抗体的编码采用实数编码, 其表示为问题 (Q) 中 p 个变量的有序排列, 抗体群的规模指定为 N , 抗体及自反应细胞对抗原的亲和力视为其对应的可行解的函数值的相反数或倒数. 借助图 4-4 的描述, 对各免疫算子作具体设计.

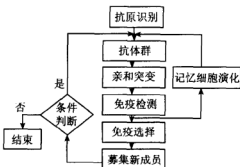


图 4-4 约束优化免疫算法的原理图

三、免疫算子设计

(1) 激励度. 设抗体群为 $P = \{x_1, x_2, \dots, x_N\}$ 及小生境半径为 σ , 设 x_i 的 σ 邻域中含 P 的 m_i 个抗体 ($i = 1, 2, \dots, N$), x_i 在 P 中的浓度取为 $C_i = \frac{m_i}{N}$, 则抗体 x_i 的激励度定义为

$$g(x_i) = \exp(-f(x_i) - KC_i),$$

其中 K 为正的调节因子, 此设计与算法 VIA 中抗体亲和力设计相似.

(2) 亲和突变. 设抗体群为 $P_k = \{x_1, x_2, \dots, x_N\}$, $x_i = (x_{i1}, x_{i2}, \dots, x_{im}), 1 \leq i \leq N$, 抗体 x_i 的突变概率选择为

$$p(x_i) = \frac{1}{1 + \exp(-\alpha f(x))}, 0 < \alpha < 1.$$

不妨设 x_1 为 P_k 中亲和力最高的抗体. 突变规则如下:

(a) P_k 中 x_1 被确定性地在其 δ 邻域产生新抗体 x'_1 , 使得 $f(x'_1) \leq f(x_1)$, 在此

$\delta = \frac{\bar{\mu}f(x_1)}{k}$, 然后 x'_1 取代 x_1 .

(b) $P_k \setminus \{x_1\}$ 中所有抗体的染色体在突变率下, 若被突变, 则按以下规则突变:

$$x'_{ij} = \begin{cases} x_{ij} + \theta \Delta_i(x_{ij}, k), & \text{rand} = 0, \\ x_{ij} - \theta \Delta_i(x_{ij}, k), & \text{rand} = 1, \end{cases}$$

$$\Delta_i(x_{ij}, k) = 1 - \left(\frac{rg(x_i)}{g_{\max}} \right)^{b(1 - \frac{k}{T})},$$

其中 b 为调节参数, 此处选择 $b=5$, T 为终止代数, r 为 $(0,1)$ 内的随机数, g_{\max} 为 P_k 中抗体的最大激励度. 在这种突变规则下, 当 x'_{ij} 不属于 $[a_j, b_j]$ 时, 其按下列规则被修正:

$$x'_{ij} = \begin{cases} a_j + \theta'(x_{ij} - a_j), & x'_{ij} < a_j, \\ x_{ij} + \theta'(b_j - x_{ij}), & x'_{ij} > b_j, \end{cases}$$

在这里, θ' 为 $[0,1]$ 上的随机数.

亲和突变体现了抗体的自适应突变及群体多样性特征, 其中邻域搜索的邻域半径动态调节, 这种策略提供了产生更好解的机会, 同时非均匀突变加大了对解空间的探测力度, 这有助于加速搜索最优解.

(3) 免疫检测. 设抗体群为 $P_k = \{x_1, x_2, \dots, x_N\}$, P_k 中亲和力最高的抗体经由邻域搜索所获新抗体不参与免疫检测, 直接参与免疫选择. 免疫检测由两部分完成. 第一部分是有效检测, 即 $P_k \setminus \{x_1\}$ 中抗体经突变后所获子代个体进行检测, 如果父代抗体突变为子代抗体, 则选择其中亲和力最高的抗体作为子代抗体. 第二部分是个体重构, 即若 $P_k \setminus \{x_1\}$ 中父代抗体突变为自反应细胞, 则对其重构. 具体而言, 设 $P_k \setminus \{x_1\}$ 经非均匀突变后变为 $P'_k = \{x'_2, \dots, x'_N\}$, 其中 x_i 变为 $x'_i (i=2, \dots, N)$, 不妨设 x'_2 为自反应细胞, x'_2 变为抗体的方法是: 利用迭代方法或随机的方法产生 t_1^* 使得 $x(t_1^*)$ 为抗体, $0 < t_1^* < 1$, 在此

$$x(t) = tx_2 + (1-t)x'_2, 0 < t < 1.$$

利用黄金分割法求 t_α , 使得

$$f(x(t_\alpha)) = \min_{t \in [t_1^*, 1]} f(x(t)),$$

所获 $x(t_\alpha)$ 作为子代抗体取代 x'_2 .

(4) 免疫选择. 这一操作采用模拟退火选择策略, 即在当前的群体 $P_k = \{a_1, a_2, \dots, a_N\}$ 中以概率

$$P(a_i) = \frac{\exp(g(a_i)/T_k)}{\sum_{j=1}^N \exp(g(a_j)/T_k)}$$

选择抗体 a_i 进入新的父代抗体群, 其中 T_k 是严格单调递减趋于 0 的退温控制序列, 通常选取 $T_k = \ln\left(\frac{T_0}{k} + 1\right)$, 其中 T_0 为初始温度。

四、约束优化免疫算法描述

利用以上所设计的算子, 并结合图 4-4, 约束优化免疫算法 COIA 可描述如下:

Step 1 若抗原为新抗原, 则在有效解集 C_{set} 中随机产生 N 个抗体构成初始群体 A_0 , 初始记忆池 M_0 为空集; 否则, 从记忆池 M_0 中随机选取部分记忆细胞及在有效集中随机选取部分抗体构成含 N 个抗体的初始群体 A_0 。

Step 2 对抗体群 A_n 进行亲和突变, 获得群体 B_n , 复制 B_n 中亲和力较高的 M 个抗体更新记忆池 M_n 。

Step 3 对 B_n 进行免疫检测, 所获得群体与经邻域搜索所获抗体组合, 获群体 C_n , 计算 C_n 中所有抗体的激励度。

Step 4 对 C_n 进行免疫选择, 获抗体群体 D_n 。

Step 5 在 C_{set} 中随机抽取 $d = \text{round}(|A_n| \cdot \mu + 1)$ 个新抗体更新 D_n 中低亲和力抗体, 获抗体群 A_{n+1} 。

Step 6 若满足终止条件, 则结束; 否则返回 Step2。

步骤 Step2 在于使算法在搜索初期抗体的突变范围变宽, 使算法搜索后期抗体突变范围变窄, 即在后期主要集中在局部搜索。Step3 在于将突变后的个体群划分为抗体群和自反应细胞群, 在所获的抗体群中, 亲和力比其母体亲和力高的抗体取代其母体参与竞争存活, 否则被清除; 对于自反应细胞群中的自反应细胞经个体重构产生的抗体参与竞争存活。Step4 使激励度较高的抗体有更大机会被选中作为下一代抗体。Step5 起到微调群体多样性的作用。该算法的特点是: (1) 群体并行搜索最优解; (2) 出现的自反应细胞被及时重构; (3) 群体对进化环境具有自适应能力; (4) 对同类问题的解决具有高效性; (5) 对待求解问题的特征信息无依赖性。

从算法 COIA 描述可看出, 有效集合 C_{set} 对算法搜索效率起重要作用, 为此以下给子生成集合 C_{set} 的算法。即

约束条件处理算法描述如下:

Step 1 将问题 (CFP) 的论域 D 等分为 2^l 个子空间, 不妨设为 $s_i, i = 1, 2, \dots, 2^l$, 初始可行解集 $C = \phi$ 。

Step 2 置 $i = 1$ 。

Step 3 若 s_i 中不包含可行解 (此可通过迭代到指定代数进行判断), 则进入 Step4; 当 s_i 中含有可行解时, 随机产生 c 个可行解, 并存入 C 中, 然后进入 Step4。

Step 4 若 $i \leq 2^l$, 则 $i \leftarrow i + 1$, 返回 Step3; 否则, $C_{set} \leftarrow C$, 结束。

注 此算法可用于非约束优化产生分布较均匀的初始群体, 此时 Step3 可改为均匀设计方案, 产生具有均匀分布的有效解集。

五、约束优化免疫算法数值实验

(一) COIA 用于约束优化问题

为了验证 COIA 解决非线性约束优化问题的有效性, 将此算法应用到例 2.5.3 中的问题 1 及问题 2, 并与算法 PA^[13] 所获结果进行比较。参数选择为, 群体规模为 80, 小生境半径 σ 为 0.3, 亲和力设计中 $K=5$, Step5 的 $\mu=0.08$ 。在这些参数下, COIA 获问题 1 的最好解为 (1.7496, 1.2384), 函数值为 $f_0 = 24.701$, 约束条件值为 $(f_1, f_2) = (0.187265, 2.7472 \times 10^{-5})$; 获问题 2 的最好解为 (622.04, 1014.86, 4557.83, 266.316, 319.152, 132.599, 334.197, 418.368), 目标函数值为 6194.73, 约束条件 $f_j, 1 \leq j \leq 6$ 的值分别为 (0.0027125, 0.0324175, 0.00784, 6089.43, 2844.71, 89.6613)。另外, 为了描述 COIA 对于解决问题 1 及问题 2 的总体行为, 任意选取该算法对上两问题分别独立运行的 10 次结果中的一次结果用图 4-5 和 4-6 描述, 图中 $\text{opt}(n)$ 及 $\text{average}(n)$ 分别表示第 n 次迭代时, $f(x)$ 在 A_n 上的最小值及平均值。

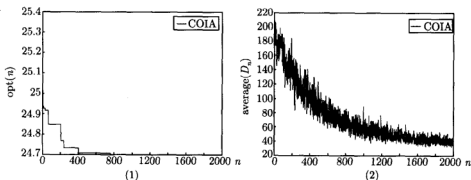
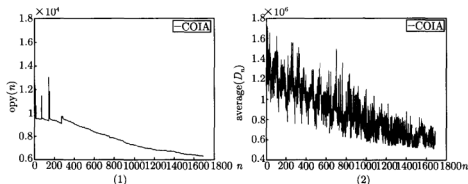


图 4-5 问题 1: (1) $f(x)$ 在 A_n 上最小值变化进程; (2) $f(x)$ 在 D_n 上平均值变化曲线

由图 4-5(1) 和 4-6 (1) 可知, COIA 对问题 1 能较快收敛到理想值, 但对于问题 2, 由于此问题的计算复杂性及约束条件的较强制约性, COIA 搜索最优解的速度稍慢, 但是所获得的实验结果 (即 6194.73) 已极大优于现有文献报道的结果 (即 7049.3307)。图 4-5(1) ~ 4-6(2) 反映了此算法的平均搜索行为, 算法搜索过程中, 进化群体的平均值处于逐步降低状态, 同时这平均值并不随迭代数的增加而恒定不变, 这说明基于抗体浓度的免疫算法能维持群体多样性。

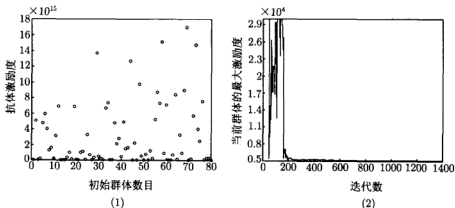

 图 4-6 问题 2: (1) $f(x)$ 在 A_n 上最小值变化进程; (2) $f(x)$ 在 D_n 上平均值变化曲线

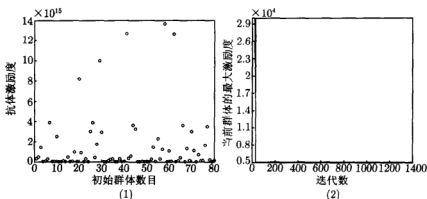
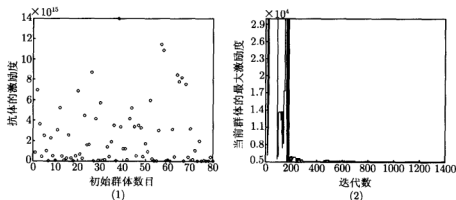
(二) 约束条件处理算法所获有效解集合 Cset 对 COIA 搜索性能的测试

从实际应用的角度, $|Cset|$ 的大小仅对算法所获最佳解的目标函数值的精度有不同程度的影响, $Cset$ 中元素的分布对收敛性无影响. 当 $|Cset|$ 在一定范围内, COIA 所获效果均相同. 以下列测试问题^[14]为例, 图 4-7 至 4-9 说明 $|Cset|$ 的不同取值及 COIA 的任意分布初始群体

$$\begin{aligned} \min f(X) &= 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3, \\ -x_3 + x_4 + 0.55 &\geq 0, x_3 - x_4 + 0.55 &\geq 0, \\ 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 &= 0, \\ 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 &= 0, \\ 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 &= 0, \\ 0 \leq x_i \leq 1200, i = 1, 2, -0.55 \leq x_i \leq 0.55, i = 3, 4 \end{aligned}$$

条件下, COIA 对该问题的搜索进程变化. 当 $|Cset|=769, 7707, 22944$ 时, COIA 搜索到最佳解的目标函数值分别为 5126.77, 5126.42, 5126.5, 此表明初始群体


 图 4-7 $|Cset|=769$: (1) 初始群体分布, (2) 搜索最优解进程

图 4-8 $|Cset|=7707$: (1) 初始群体分布, (2) 搜索最优解图 4-9 $|Cset|=22944$: (1) 初始群体分布, (2) 搜索最优解进程

及 $|Cset|$ 的取值仅影响算法搜索最优解的精度, 当 $|Cset| \geq 800$ 时, $|Cset|$ 的不同取值对 COIA 搜索效果基本不影响, 这表明本文的约束条件处理算法能提高 COIA 的搜索性能。

第五节 模糊控制免疫算法

一、预备知识

模糊集的概念由扎得于 1965 年提出, 至今已形成独立的数学分支, 即模糊数学, 其已在各研究领域及工程领域获得广泛应用。根据这部分及第五章的需要, 以下简要介绍模糊集及相关概念和性质, 详细内容可参阅文献 [15, 16]。

(1) 模糊集。设在 U 上给定映射 $\mu, \mu: U \rightarrow [0, 1]$, 称 μ 确定了 U 上的一个模糊集合 \tilde{A} , μ 称为 \tilde{A} 的隶属度函数, 记作 $\mu_{\tilde{A}}, \mu_{\tilde{A}}(u)$ 称为 u 对 \tilde{A} 的隶属度, 它表示 u 属于 \tilde{A} 的程度。 U 上的模糊集合 \tilde{A} 简称为模糊集。

(2) 贴近度. 设 $F(U)$ 为论域 U 上的所有模糊集构成的集合, 映射 $N: F(U) \times F(U) \rightarrow [0, 1]$ 称为 U 上的模糊集与模糊集之间的贴近度函数, 其若满足:

- (a) $N(\bar{A}, \bar{B}) = N(\bar{B}, \bar{A})$;
- (b) $N(\bar{A}, \bar{A}) = 1$;
- (c) $\bar{C} \subseteq \bar{B} \subseteq \bar{A} \Rightarrow N(\bar{A}, \bar{C}) \leq N(\bar{A}, \bar{B}) \wedge N(\bar{B}, \bar{C})$;

则函数值 $N(\bar{A}, \bar{B})$ 称为模糊集 \bar{A} 与 \bar{B} 的贴近度.

(3) 最大隶属度原则. 设论域 U 中有 n 个模糊集 $\bar{A}_1, \bar{A}_2, \dots, \bar{A}_n, \mu_{\bar{A}_i}(x)$ 为 \bar{A}_i 的隶属度函数, 若成立

$$\mu_{A_j}(x_0) = \max\{\mu_{\bar{A}_i}(x_0), 1 \leq i \leq n\},$$

则 $x_0 \in \bar{A}_j$.

(4) 模糊联结基函数. 模糊联结基函数是 F. Herrera 于 1997 年首次提出^[17], 并在此基础上提出了模糊遗传算法的概念. 其通过设计模糊遗传算子及模糊规则 (用于调整交叉概率和突变概率), 建立了模糊遗传算法. 此算法的关键在于构建模糊交叉算子, 其方法如下:

设 $x = (x_1, x_2, \dots, x_n), y = (y_1, y_2, \dots, y_n)$ 表示两个个体, 个体中的元素 x_i, y_i 称为染色体, $x_i, y_i \in [a_i, b_i]$. x_i, y_i 将 $[a_i, b_i]$ 区间划分为三个区间, 如图 4-10 所示. 根据区间 $[a, x_i], [x_i, y_i], [y_i, b]$ 的开

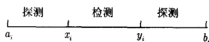


图 4-10 区间划分

采与探测的程度不同, 可设计三个函数, 即模糊联结基函数, 用于度量这三个区间的不同重要程度, 两染色体 x_i, y_i 利用此三个函数可获三个不同的值, 即子代的染色体. 进而个体 x, y 交配所获个体是由各对染色体作用模糊联结基函数获的染色体组成, 从而获子代个体. 这种个体之间利用模糊联结基函数进行交配的方式可产生大量的子代个体, 增强了算法探测能力和群体多样性. 以下介绍几种类型的模糊联结基函数^[18], 其中 $0 \leq \lambda \leq 1$.

类型 1 $T_L(x, y) = \min(x, y), G_L(x, y) = \max(x, y), P_L(x, y) = (1 - \lambda)x + \lambda y$;

类型 2 $T_H(x, y) = \frac{xy}{x + y - xy}, G_H(x, y) = \frac{x + y - 2xy}{1 - xy}$,

$$P_H(x, y) = \left(\frac{(1 - \lambda)y - (y - \lambda)x}{xy} + 1 \right)^{-1};$$

类型 3 $T_A(x, y) = xy, G_A(x, y) = x + y - xy, P_A(x, y) = x^{1-\lambda}y^\lambda$;

类型 4 $T_E(x, y) = \frac{xy}{1 + (1 - x)(1 - y)}, G_E(x, y) = \frac{x + y}{1 + xy}$,

$$P_E(x, y) = 2 \left(1 + \left(\frac{2-x}{x} \right)^{1-\lambda} \left(\frac{2-y}{y} \right)^{\lambda} \right)^{-1}.$$

这些模糊连结基函数可混合使用, 两个体的染色体可通过此表中任一函数确定子代的染色体. 这种利用模糊联结基函数设计交叉算子的方法启发我们利用这些基函数设计免疫算法中的亲和突变算子, 增强算法的局部搜索能力.

二、模糊控制免疫算法描述

利用体液免疫应答过程的部分机理设计免疫算子, 其中亲和突变算子是利用以上模糊连结基函数设计, 同时建立模糊控制参数调节系统, 动态调节算法中的参数. 将免疫算子及模糊控制参数调节系统有机组合, 则可建立具有动态调节群体规模和参数的模糊控制免疫算法 (FCIA: Fuzzy Control Immune Algorithm), 并用于函数优化问题 (NFP), 尤其是对多模态函数优化的应用.

在算法设计中, 抗体与抗原均按本章引言所述的实数编码表示, 抗体视为 NFP 的可行解, 抗原视为算法搜索的当前群体中的最好解, 也就是说, 当前群体中最好抗体具有两重身份, 一是作为被识别的抗原, 二是与其他个体一样作为抗体. 抗体 Ab 的激励度定义为

$$g(Ab) = \frac{1}{1 + \exp(\hat{\mu} \cdot f(Ab))},$$

其中 $\hat{\mu}$ 为可调参数, $f(Ab)$ 为抗体 Ab 所对应的可行解的目标值. 抗体 Ab 与抗原 Ag 的亲和力定义为

$$aff(Ab) = \frac{1}{1 + \exp(\|Ab - Ag\|)}. \quad (4.5.1)$$

于是模糊免疫算法描述如下:

Step 1 随机产生规模为 \hat{N} 的初始抗体群 A_1 , 设置初始参数 $\sigma_i, i = 1, 2, 3, 4$ 及确定进化代数 $n \leftarrow 1$.

Step 2 计算 A_n 中抗体的激励度, A_n 中激励度最高的抗体作为抗原 Ag_n .

Step 3 利用模糊控制参数调节系统确定参数 $\sigma_i, i = 1, 2, 3, 4$.

Step 4 将 A_n 依据

$$\|Ab_i - Ab_j\| < \sigma_1, Ab_i, Ab_j \in A_n$$

划分为互不相交的 q 个子群 Q_1, Q_2, \dots, Q_q , 选择各子群中激励度最高的抗体构成记忆细胞群 M_n .

Step 5 依据式 (5.5.1), 计算 A_n 中各抗体与抗原 Ag_n 的亲和力, 选择 $N_n \equiv \text{round}(\sigma_2 |A_n|)$ 个抗体构成群体 B_n .

Step 6 B_n 中每一抗体 Ab 被克隆三个克隆细胞 $Ab_i = (x_{i1}, x_{i2}, \dots, x_{ip}), i = 1, 2, 3$, 此三个细胞被突变的概率设计为

$$\lambda = \frac{\mu_n + \max_{z \in A_n} f(z) - f(Ab)}{\mu_n + \max_{z \in A_n} f(z) - \min_{z \in A_n} f(z)}, \mu_n = \mu + \frac{T_0}{n},$$

其中, μ, T_0 为正的可调参数, $0 < \mu < \frac{1}{2}$, 不妨记 $Ag_n = (y_{n1}, y_{n2}, \dots, y_{np}), Ab_1$ 中的染色体 x_{1j} 的突变方式为

$$x'_{1j} = T_E(x_{1j}, y_{nj}), j = 1, 2, \dots, p.$$

Ab_2 中的染色体 x_{2j} 的突变方式为

$$x'_{2j} = G_E(x_{2j}, y_{nj}), j = 1, 2, \dots, p.$$

Ab_3 中的染色体 x_{3j} 的突变方式为

$$x'_{3j} = P_L(x_{2j}, y_{nj}), j = 1, 2, \dots, p.$$

按此方式获 $3N_n$ 个突变的克隆细胞构成群体 C_n .

Step 7 消除 C_n 中相同的克隆, 并根据

$$\|Ag - Ab_i\| < \sigma_3, Ab_i \in C_n$$

消除低亲和力的克隆, 获存活的克隆群体 D_n .

Step 8 将群体 M_n 与 D_n 组合, 同时随机产生 $\text{round}(\sigma_4|A_n|)$ 个新抗体取代 $M_n \cup D_n$ 中低亲和力的抗体, 获新抗体群 A_{n+1} .

Step 9 终止条件判断, 若满足条件, 则结束; 否则, 返回 Step2.

上算法中, 进化群体规模及参数 $\sigma_i, i = 1, 2, 3, 4$ 具有相互协调、相互抑制和促进的作用, 这些参数控制该算法的搜索行为, 群体 A_n 的规模在这些参数的作用下进行扩大或缩小, σ_1 决定记忆细胞保存数目, σ_2 控制被选择进行克隆的抗体规模, σ_3 起到抑制相似克隆, 防止相似克隆被传递到新抗体群的作用, σ_4 主要在于增强群体的多样性. 将进化群体的规模作为输入, 参数 $\sigma_i, i = 1, 2, 3, 4$ 作为输出, 通过设计模糊规则并建立参数模糊控制系统动态调节群体规模.

三、参数调节模糊控制系统

(一) 模糊输入变量

模糊输入变量选定为算法 FCIA 中进化群体 A_n 的规模, 不妨记 $N \equiv |A_n|$, 则 N 为可变量, 其语言变量集设为 { 小, 中, 大 }, 如图 4-11 所示.

(二) 模糊输出变量

模糊输出变量设定为 σ_2 以及变量 $\sigma_i, 1 \leq i \leq 4$ 的改变量 $\delta\sigma_i, 1 \leq i \leq 4$. 输出变量 $\sigma_i, 1 \leq i \leq 4$ 的论域设定为 $\sigma_i \in [0, 1], i = 1, 2, 3, 4$. 这些变量值的确定是 $\sigma_i \leftarrow \sigma_i \cdot \delta\sigma_i$, 即 σ_i 的前一值与其改变量 $\delta\sigma_i$ 之积, 其中 σ_2 既作为模糊变量又作为输出变量. $\delta\sigma_i, 1 \leq i \leq 4$ 的论域设定为

$$\delta\sigma_1 \in [0, 1.2], \delta\sigma_2 \in [0, 1.7], \delta\sigma_3 \in [0, 1.2], \delta\sigma_4 \in [0, 1.2],$$

σ_2 的语言变量集为 { 小, 大 }, 其各语言变量的含义如图 4-12 所示.

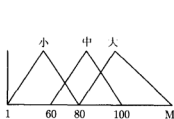


图 4-11 N 的语言变量的物理意义

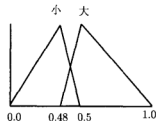


图 4-12 σ_2 的语言变量的物理意义

变量 $\delta\sigma_1, \delta\sigma_2, \delta\sigma_3, \delta\sigma_4$ 的语言变量集分别为 { 低, 中, 高 }, { 低, 高 }, { 低, 中, 高 } 及 { 低, 高 }, 各变量的语言变量的意义如图 4-13 所示.

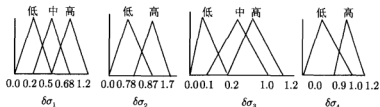


图 4-13 $\delta\sigma_i, 1 \leq i \leq 4$ 的语言变量的物理意义

根据各变量的语言变量的含义, 以下建立模糊规则调节算法 FICA 中各参数

规则 1 如果 N 小, 那么 $\delta\sigma_1$ 低, $\delta\sigma_2$ 高;

规则 2 如果 N 大, 那么 $\delta\sigma_1$ 高, $\delta\sigma_2$ 低;

规则 3 如果 σ_2 小, 那么 $\delta\sigma_3$ 高, $\delta\sigma_4$ 低;

规则 4 如果 σ_2 大, 那么 $\delta\sigma_3$ 低, $\delta\sigma_4$ 高.

四、模糊控制免疫算法仿真分析

模糊免疫控制算法 (FCIA) 是一种动态调整群体规模和参数的新智能算法. 为了能更清楚地获知该算法解决实际问题所表现出的行为特性, 现将此算法用于例

2.5.2 中测试函数 F_1 、 F_3 、 F_4 求最小值。测试内容包括参数的变化, 进化群体规模的变化, 搜索最优解的进程和精度分析, 以及算法搜索的突变性质分析。该算法的初始参数选择为: $\sigma_1=0.20$, $\sigma_2=0.615$, $\sigma_3=0.56$, $\sigma_4=0.545$, 初始群体规模为 120。对于参数 σ_i , $1 \leq i \leq 4$ 的变化情况, 仅用该算法作用于测试函数 F_4 的一次运行结果加以说明, 如图 4-14(c)、(d)、(e)、(f)。用 n 表示迭代次数, $z[n]$ 表示 $f(x)$ 在 An 上的最小值, $|An|$ 表示群体 An 的规模, $niche[n]$ 、 $clone[n]$ 、 $negative[n]$ 、 $news[n]$ 分别表示算法在第 n 代的算法中参数 $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ 随 n 的变化值。现对该算法作用于所列举的三个测试函数的一次运行结果用图 4-14~4-16 描述。

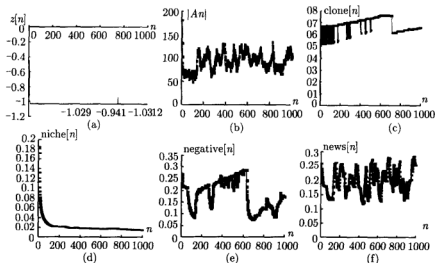


图 4-14 函数 F_4 : (a) 搜索最优解的变化曲线; (b) 进化群体规模的变化曲线; (c)、(d)、(e)、(f) 分别为参数 $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ 的变化曲线

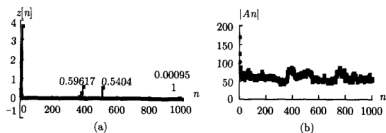


图 4-15 对于函数 F_1 : (a) 最优解搜索进程曲线; (b) 群体规模变化曲线

(1) 算法 FCIA 用于函数 F_4 的一次运行结果如图 4-14 所示。

在算法设计中, $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ 随迭代数的增加而作自适应调整, 参数 σ_1, σ_3 与进化群体的规模 N 成正比, σ_2, σ_4 与 N 成反比, σ_2 与 σ_3 成反比。从图 4-14 的图 (c)、(d)、(e)、(f) 可看出, 这些参数的设计与算法用于测试问题时, 所表现

的行为刚好一致。特别在算法搜索后期,为了适当增大 N , σ_1 , σ_3 逐渐有所减小,导致 σ_2 稍微变大,进而获 N 在合适范围内,尤其 σ_4 对 N 的扩大和缩小有重要作用。在搜索后期, σ_4 有所增大,进而使 N 增大, σ_1 减小, σ_3 增大,这有助于优良抗体的保存和群体多样性的维持,因此这四个参数相互协调和相互抑制,使进化群体的多样性及规模在确定范围变化得以保证。

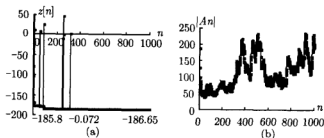


图 4-16 对于函数 F_3 : (a) 最优解搜索进程曲线; (b) 群体规模变化曲线

(2) 将算法 FCIA 分别用于函数 F_1 、 F_3 的一次运行结果用图描述,如图 4-15 及图 4-16。

从图 4-14~4-16(a) 可知,算法搜索 F_1 、 F_3 、 F_4 所获实验解的目标值分别为 0.000951, -186.653, -1.03126,这与各函数的理想目标值的偏差分别为 0.000951, 0.077, 0.000368。由此可见,若忽略计算误差,则该算法所搜索的实验解即为理想解,这也反映了通过模糊控制系统调节参数对系统行为具有重要作用。

从图 4-14(b)~4-16(b) 可看出,算法在搜索过程中, N 始终处于不断变化中,其变化范围视各测试函数而定。对于 F_1 、 F_3 、 F_4 ,群体变化范围分别是 [48,100], [48,210] 及 [50,140],这反映了算法中的各参数及群体的规模相互依存及相互抑制,共同促进群体多样性及最优解的搜索进程。

另外,参数的自动调节对于进化群体跳出局部最优解所在区域至关重要,这可从图 4-14(a)~4-16(a) 获得启示。对于图 4-14(a),算法用于 F_4 ,在第 751 代之前已搜索到次优解 -1.02968,在第 751 代时,进化群体的所有抗体中最小值却为 -0.94107,这表明算法中的参数处于较大调整,促成群体搜索更好解,即到第 751 代之后,算法已获得几乎为理想解的解。同理,对于 F_1 、 F_3 ,由图 4-15~4-16,算法分别在第 9 代及第 259 代(和第 319 代)发生突跃,促成群体跳出局部搜索区域,因而该算法具有突现性质。

综合以上分析可得出,模糊控制免疫算法是一种自组织搜索最优解的新智能算法,从算法中的参数变化趋势可知,参数的变化导致加速最优解的搜索。特别在搜索后期,参数增强了群体多样性,以及群体规模在恒定范围内不断变化,同时参数的变化使算法有较好的突现性质,因此该算法对实际问题的解决有一定的借鉴作用。

第六节 形态空间上免疫算法的收敛性

在这一部分,对以上所获的几种形态空间上的免疫算法的收敛性进行理论研究.由于抗体由 p 个实变量作为染色体构成,不妨设每一染色体由 l 个 0 到 9 之间的整数作为基因构成,因此所有抗体构成有限的抗体空间 S , 设 $S^{\leq N}$ 表示群体规模不超过 N 的非空抗体群构成的抗体群空间, S^N 由群体规模为 N 的抗体群构成的抗体群空间. 将 S^N 及 $S^{\leq N}$ 中每一个元素表示为状态 s_{i_1}, s_{j_1}, \dots . 符号 M^* 的含义及算法的收敛性定义与第三章的描述相同,在此省略. 用 A_n^i 表示 n 时刻 A_n 处于状态 s_i , $P(A_n^i)$ 表示 A_n 处于状态 s_i 的概率, $P(A_{n+1}^j | A_n^i)$ 指在 A_n 处于 s_i 下, A_{n+1} 处于状态 s_j 的概率. 用 Max 及 Min 分别表示 $f(x)$ 在 S 中所取的最大和最小值, $T^d(S)$ 表示随机产生的 d 个新抗体构成的集合, $P(T^d(S) = s_j)$ 表示状态为 s_j 的概率, $P(A_n \cap M^* \neq \emptyset)$ 表示 n 时刻 A_n 与 M^* 的交集为非空的概率.

一、小生境免疫算法的收敛性

由于小生境免疫算法 NIA 中 A_{n+1} 所处状态仅与 A_n 及 D_n^d 有关, 与 n 时刻以前的状态无关, 故由链

$$A_n \rightarrow B_n \cup M_n \rightarrow C_n \cup M_n \cup D_n^d \rightarrow A_{n+1}$$

描述的随机搜索过程为马氏链, 其中 D_n^d 表示 n 时刻随机产生的 d 个新抗体, 但由于亲和突变率与时间 n 有关, 故该链为非齐次马氏链. 于是引入如下定理.

定理 4.6.1 算法 NIA 对于任意初始分布, 均以概率弱收敛.

证明 由于第四章第二节的共享机制算法及算法 NIA 的克隆选择皆为确定性的操作, 因此若 A_n 处于状态 s_i , 则不妨设经共享机制算法所获状态为 s_{i_1} , 经克隆选择所获状态为 s_{i_2} , 从而

$$P(M_n^{i_1} | A_n^i) = P(B_n^{i_2} | A_n^i) = 1, s_{i_1} \in S^M, s_{i_2} \in S^{N_0}. \quad (4.6.1)$$

由算法 NIA 的步骤 4 知, 对于任意 $s_k, s_s \in S^{N_0}$, 若用 $P(x_{nk_t} \rightarrow y_{ns_t})$ 表示 n 时刻状态 s_k 中第 t 个分量 x_{nk_t} 变为状态 s_s 中第 t 个分量 y_{ns_t} 的概率, 则有

$$P(C_n^s | B_n^k) = \prod_{t=1}^{N_0} P(x_{nk_t} \rightarrow y_{ns_t}) \geq \prod_{t=1}^{N_0} \left(\left(\frac{9}{100} \right)^{p_t} (1 - \exp(-p_{nt})) \right), \quad (4.6.2)$$

其中 $p_{nt} = \frac{\mu_n + \max_n f(x_{nk_t})}{\mu_n + \max_n - \min_n}$, 由泰勒展开式定理知

$$\begin{aligned}
1 - \exp(-p_{nt}) &= p_{nt} \left(1 - \frac{1}{2} \exp(-\theta \cdot p_{nt}) p_{nt} \right) \\
&\geq \frac{\mu_n}{2(\mu_n + \max_n - \min_n)} \geq \frac{\mu}{2(\mu + \text{Max} - \text{Min})}, \quad 0 < \theta < 1.
\end{aligned} \tag{4.6.3}$$

进而由式 (4.6.2) 及式 (4.6.3) 知

$$P(C_n^s | B_n^k) \geq \left(\left(\frac{9}{100} \right)^{Pl} \frac{\mu}{2(\mu + \text{Max} - \text{Min})} \right)^{N_0}. \tag{4.6.4}$$

现考虑 s_i 转移为 s_j 的概率. 由步 5 知, 若 A_{n+1} 处于状态 s_j , 则此状态由三部分构成, 一部分由 s_i 经共享机制算法所获的状态 s_{j2} , 另一部分为 s_i 经步 2 至步 4 转移所获状态 s_{j1} , 其余部分为随机产生的 d 个新抗体构成的状态 s_{j3} , 于是由式 (4.6.1)、式 (4.6.4) 及步 5 知

$$\begin{aligned}
P(A_{n+1}^j | A_n^i) &= P(A_{n+1}^{j_1, j_2} | A_n^i) P(T^d(S)) \\
&= \sum_{s_k \in S^{N_0}} P(C_n^k | B_n^{i_2}) P(A_{n+1}^{j_1} | M_n^{i_1}) P(T^d(S) = s_{j_3}) \\
&\geq \left(\frac{\mu}{2 \times 10^{Pl} (\mu + \text{Max} - \text{Min})} \right)^{N_0} P(T^d(S) = s_{j_3}).
\end{aligned} \tag{4.6.5}$$

对于 $s_{j_3} \in S^d, s_{j_3} \subset M^*$, 必有

$$P(T^d(S) = s_{j_3}) = \left(\frac{|M^*|}{|S|} \right)^d.$$

进而由式 (4.6.5) 获知, 对于 $s_{j_0} \subset M^*$,

$$\begin{aligned}
&\sum_{s_i \cap M^* = \phi} P(A_{n+1} \cap M^* = \phi | A_n^i) P(A_n^i) = \sum_{s_i \cap M^* = \phi} (1 - P(A_{n+1} \cap M^* \neq \phi | A_n^i)) P(A_n^i) \\
&\leq \sum_{s_i \cap M^* = \phi} \left(1 - P(A_{n+1}^{j_0} | A_n^i) \right) P(A_n^i) \leq \left(1 - \left(\frac{|M^*|}{|S|} \right)^d \left(\frac{\mu}{2 \times 10^{Pl} (\mu + \max - \min)} \right)^{N_0} \right) \\
&\quad P(A_n \cap M^* = \phi).
\end{aligned} \tag{4.6.6}$$

另外, 由共享机制算法及算法 NIA 的步骤 5 知, A_n 所处的状态中的最优抗体被保存到 A_{n+1} 所处的状态中, 因此当 $s_i \cap M^* \neq \phi$ 时,

$$\sum_{s_i \cap M^* \neq \phi} P(A_{n+1} \cap M^* = \phi | A_n^i) P(A_n^i) = 0,$$

故由全概率公式知

$$\begin{aligned} a_{n+1} &\equiv P(A_{n+1} \cap M = \phi) = \left(\sum_{s_i \cap M^* = \phi} + \sum_{s_i \cap M^* \neq \phi} \right) P(A_{n+1} \cap M^* = \phi | A_n^i) P(A_n^i) \\ &= \sum_{s_i \cap M^* = \phi} P(A_{n+1} \cap M^* = \phi | A_n^i) P(A_n^i). \end{aligned} \quad (4.6.7)$$

进而由式 (4.6.6) 及式 (4.6.7) 获

$$a_{n+1} \leq \left(1 - \left(\frac{|M^*|}{|S|} \right)^d \left(\left(\frac{9}{100} \right)^{p_l} \frac{\mu}{2(\mu + \max - \min)} \right)^{N_0} \right) a_n.$$

从而 $\lim_{n \rightarrow \infty} a_n = 0$, 此表明该定理的结论成立.

二、动态免疫算法的收敛性

由算法 DIA 描述获知, 该算法可描述为齐次马氏链. A_{n+1} 由三部分构成, 即一部分为被选择参与进化所获的抗体群, 另一部分为未被选中参与进化的抗体经过步 5 作用后所获抗体群 (规模设为 $S(A_n, \alpha, \sigma)$), 其余部分为随机产生的新抗体 (规模为 $\text{round}(\mu|A_n|)$), 于是 A_{n+1} 的群体规模为

$$|A_{n+1}| = \text{round}((\alpha + \mu)|A_n|) + S(A_n, \alpha, \sigma). \quad (4.6.8)$$

另外, 由于 σ 依据群体规模 $|A_n|$ 自适应调整, 因此 $S(A_n, \alpha, \sigma)$ 的大小由 α, σ 确定, 而 σ 与 $|A_n|$ 成正比, 因此 $S(A_n, \alpha, \sigma)$ 能被动态调节.

定理 4.6.2 若 $S(A_n, \alpha, \sigma) \leq M, \alpha + \mu < 1$, 则算法 DIA 的群体规模满足

$$|A_n| \leq |A_0| + \frac{1}{1 - (\alpha + \mu)} M.$$

证明 由式 (4.6.8) 及步 2 至步 6 知,

$$\begin{aligned} |A_{n+1}| &= \text{round}((\alpha + \mu)|A_n|) + S(A_n, \alpha, \mu) \\ &\leq (\alpha + \mu)|A_n| + S(A_n, \alpha, \mu), \end{aligned} \quad (4.6.9)$$

则式 (4.6.9) 通过归纳获知

$$\begin{aligned} |A_{n+1}| &= (\alpha + \mu)^{n+1}|A_0| + \sum_{k=0}^n (\alpha + \mu)^k S(A_{n-k}, \alpha, \mu) \\ &\leq (\alpha + \mu)^{n+1}|A_0| + M \sum_{k=0}^n (\alpha + \mu)^k \leq |A_0| + \frac{1}{1 - (\alpha + \mu)} M \equiv \hat{M}. \end{aligned}$$

故该定理的结论成立. 证毕.

定理 4.6.3 对于任意初始分布, 动态免疫算法在定理 4.6.2 的条件下是概率弱收敛.

证明 由定理 4.6.2 知, 该算法的进化群体的规模始终不超过 \hat{M} . 由该算法中步 4 的突变方式获知

$$\min f(A_{n+1}) \leq \min f(A_n), n = 1, 2, \dots, n.$$

这表明

$$P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) = 0. \quad (4.6.10)$$

另外

$$\begin{aligned} P(T^{d_n}(S) \cap M^* \neq \phi) &= 1 - P(T^{d_n}(S) \cap M^* = \phi) \\ &= 1 - \left(\frac{|S| - |M^*|}{|S|} \right)^{d_n} \geq 1 - \left(\frac{|S| - |M^*|}{|S|} \right)^{\hat{M}} \equiv \delta, \end{aligned} \quad (4.6.11)$$

则由式 (4.6.11) 及步 6 可获

$$P(A_{n+1} \cap M^* \neq \phi | A_n \cap M^* = \phi) \geq \delta, \quad (4.6.12)$$

从而由式 (4.6.10)、式 (4.6.12) 及定理 3.6.1 可知该定理的结论成立. 证毕.

三、约束优化免疫算法的收敛性

由于约束优化免疫算法中 A_{n+1} 所处状态仅与 A_n 所处状态有关, 与过去状态无关, 记忆池仅对同类优化问题的解决时, 对初始群体的选择方面有作用, 对当前问题的解决无任何影响. 故由链

$$A_n \rightarrow B_n \rightarrow C_n \rightarrow D_n \rightarrow A_{n+1}$$

描述的随机搜索过程为马尔可夫链, 但由于亲和突变及免疫选择与时间有关, 故该链为非齐次马尔可夫链.

定理 4.6.4 约束优化免疫算法 COIA 对于任意初始分布均概率弱收敛.

证明 不妨假设 M^* 仅包含一个元素. 由 COIA 中步骤 4 和 5 知, 当 A_{n+1} 处于状态 s_j 时, s_j 由步骤 4 所获 $N-d$ 个抗体 $s_{j_1} = (x_1^j, x_2^j, \dots, x_{N-d}^j)$ 以及步骤 5 随机产生的 d 个新抗体 $s_{j_2} = (x_1^j, x_2^j, \dots, x_d^j)$ 构成. 由免疫检测算子的设计获知, 当 $A_n = s_i, s_i \cap M^* \neq \phi$ 时, 对于 $x \in s_i \cap M^*$, 经非均匀突变 x 变为 x' , 若 x' 为自反应细胞, 则经黄金分割法所获解必为最优解. 因此

$$P(C_n^c | A_n^i) = \begin{cases} 1, & s_i \cap M^* \neq \phi, s_c \cap M^* \neq \phi, \\ 0, & s_i \cap M^* \neq \phi, s_c \cap M^* = \phi, \end{cases} \quad (4.6.13)$$

从而由全概率及免疫算子知, 当 $s_i \cap M^* \neq \phi$,

$$P(A_{n+1}^j | A_n^i) = \sum_{s_c \cap M^* \neq \phi} P(C_n^c | A_n^i) P(A_{n+1}^j | C_n^c), \quad (4.6.14)$$

其中

$$P(A_{n+1}^j | A_n^i) = P(A_{n+1}^{j_1} | C_n^c) P(T_d(S) = s_{j_2}),$$

$$P(A_{n+1}^{j_1} | C_n^c) = \begin{cases} \prod_{m=1}^N \frac{\exp(g(x_m)/T_k)}{\sum_{c=1}^N \exp(g(x_c)/T_k)}, & s_{j_1} \subset s_c, \\ 0, & \text{其他.} \end{cases} \quad (4.6.15)$$

(a) 当 $s_i \cap M^* \neq \phi$, $s_j \cap M^* = \phi$ 时, 对任意 $x_0 \in s_{j_1}, x^* \in s_i \cap M^*$, 用 C^* 及 C_0 分别表示抗体 x^* 及 x_0 在所处群体中的浓度, 令

$$\rho_0 = \min \{|f(x^*) - f(y)|, f(x) \neq f(y), y \in S\}, \quad (4.6.16)$$

从而由式 (4.6.15) 获

$$P(A_{n+1}^{j_1} | C_n^c) \leq \frac{\exp(g(x_0)/T_n)}{\exp(g(x^*)/T_n)} = \exp(g(x_0) - g(x^*)/T_n), \quad (4.6.17)$$

其中

$$\begin{aligned} & g(x^*) - g(x_0) \\ &= \exp(-f(x^*) - KC^*) - \exp(-f(x_0) - KC_0) \\ &= \exp(-f(x_0) - KC_0)(\exp(f(x_0) - f(x^*) + K(C_0 - C^*)) - 1) \\ &\geq \exp(-f_{\max} - K)(\exp(\rho_0 - K) - 1) \equiv \delta_0. \end{aligned} \quad (4.6.18)$$

于是由式 (4.6.14)、(4.6.17)、(4.6.18) 获

$$P(A_{n+1}^{j_1} | A_n^i) \leq \exp(-\delta_0/T_n). \quad (4.6.19)$$

又

$$(T_d(S) = s_{j_2}) = \left(\frac{m - m^*}{m^2} \right)^d,$$

这里, $m = |\text{Cset}|$, $m^* = |C^*\text{set}|$, $C^*\text{set} = \text{Cset} \cap M^*$, 从而

$$P(A_{n+1}^j | A_n^i) \leq |S^N| \exp(-\delta_0/T_n) \left(\frac{m - m^*}{m^2} \right)^d = \varepsilon_n.$$

(b) 当 $s_i \cap M^* = \phi, s_{j_1} \subset M^*$ 时, 记

$$\begin{aligned} M_1(j_1) &= (A_{n+1} = s_{j_1}, s_{j_1} \subset M^*), \\ M_2(j_2) &= (T^d(S) = s_{j_2}, s_{j_2} \subset \text{Cset}), \end{aligned}$$

则有

$$P(M_1(j_1)|A_n^i) \geq \left(\frac{1}{N}\right)^{N-d} = \delta_1.$$

又因 d 个新抗体于 Cset 中随机性产生, 则有

$$P(M_2(j_2)|A_n^i) = \left(\frac{m^*}{m^2}\right)^d \equiv \delta_2,$$

从而

$$P(A_{n+1} \cap M^* \neq \phi | A_n^i) = \max(P(M_1(j_1)|A_n^i), P(M_2(j_2)|A_n^i)) = \delta.$$

在这里, $\delta = \max(\delta_1, \delta_2)$. 进而

$$\begin{aligned} P(A_{n+1} \cap M^* = \phi) &= \left(\sum_{s_i \cap M^* = \phi, s_j \cap M^* \neq \phi} + \sum_{s_i \cap M^* \neq \phi, s_j \cap M^* = \phi} \right) P(A_{n+1}^j | A_n^i) P(A_n^i) \\ &\leq (1 - \delta) P(A_n \cap M^* = \phi) + |S^N|^2 \varepsilon_n. \end{aligned}$$

又 $0 < \delta < 1$, $\sum_{k=1}^{\infty} \varepsilon_k < \infty$, 则有

$$\lim_{n \rightarrow \infty} P(A_n \cap M^* = \phi) = 0.$$

故结论成立.

从以上证明获知, 约束条件处理算法所获的约束集的元素的分布及所包含的数目对算法的收敛性无影响.

第七节 应用举例

形态空间上的免疫算法通过对多峰值函数优化的应用, 获得其用于函数优化具有较好的搜索性能. 在此以小生境免疫算法 (NIA) 及动态规模免疫算法 (DIA) 分别解决磁盘驱动器的磁头臂控制及定常连续线性系统的极点配置问题为例, 说明此类形态空间上的免疫算法解决实际问题的有效性.

例 4.7.1 磁头臂控制系统参数辨识问题.

(一) 问题的提出

磁盘驱动器控制问题是一种典型的伺服控制问题, 控制的目的是使磁头臂沿着指定的轨迹运动, 其原理如图 4-17 所示. 其中 u_c 为指定的参考输入, y 为磁头臂位置输出. 控制器输出 u 到磁头臂位置输出 y 的传递函数可近似描述为

$$G(s) = \frac{k}{Js^2}, \quad (4.7.1)$$



图 4-17 磁盘驱动器系统

磁头控制器的数学模型可简单描述为

$$U(s) = \frac{bK}{a}U_c(s) - K\frac{s+b}{s+a}Y(s), \quad (4.7.2)$$

其中 J 为臂组件的惯性矩, a, b, k, K 为参数. 这些参数对系统在指定参考输入下磁头臂的运动具有不同程度影响, 式 (4.7.1)、(4.7.2) 可由文献 [18] 获知. 为了使磁头臂沿指定轨迹运动, 在此利用算法 NIA 确定这些参数的最佳取值. 由式 (4.7.1) 及式 (4.7.2) 获输入 u_c 到输出 y 的传递函数有如下关系:

$$Y(s) = \frac{kKb(s+a)}{a(Js^3 + aJs^2 + kKs + kKb)}U_c(s). \quad (4.7.3)$$

由式 (4.7.3) 可获磁盘驱动器系统的状态空间描述, 即

$$(CS) \quad \dot{X} = AX + Bu_c, \quad y = CX, \quad (4.7.4)$$

其中

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{kKb}{J} & -\frac{kK}{J} & -a \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} akKb & kKb & 0 \end{bmatrix}.$$

由于参数 a, b, k, K, J 皆不为 0, 因而 A 可逆, 从而系统 (4.7.4) 的离散化模型为

$$(DS) \quad \begin{aligned} X(t+h) &= e^{Ah}X(t) + A^{-1}(E - e^{Ah})Bu_c(t), \\ y(t+h) &= CX(t+h), \end{aligned} \quad (4.7.5)$$

其中, h 为采样周期. 设 P 为采样数目, 即时间 $T = P \cdot h$. 选择目标函数为均方差函数, 即

$$\bar{J}(a, b, k, K, J) = \frac{1}{2} \sum_{k=1}^P (y(kh) - d(kh))^2,$$

其中 $d(x)$ 为期望输出函数. 在此选择

$$d(x) = 1 - (1+x)e^{-x},$$

于是确定磁盘驱动器系统最优参数的最小化问题可描述为

$$(P) \quad \begin{aligned} \min \bar{J}(a, b, k, K, J), \\ (a, b, k, K, J) \in \mathbb{R}^5. \end{aligned}$$

(二) 算法 NIA 与磁盘驱动器系统的衔接及仿真结果

将问题 (P) 中变量组 (a, b, k, K, J) 视为抗体, 目标函数 \bar{J} 作为抗原, 利用免疫算法 NIA 对磁盘驱动器系统的参数优化. 由于该算法在优化参数过程中, 抗体不断被评价, 因而磁盘驱动器系统将不断重复采样, 此算法与磁盘驱动器系统的关系如图 4-18 所示.

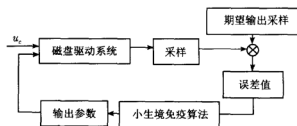


图 4-18 算法 NIA 与磁盘驱动器系统关系

由于算法 NIA 为基于群体搜索的优化算法, 因此算法每迭代一次能获得较多的抗体 (即磁盘驱动器系统的参数组), 从而通过算法搜索, 最终能获得该系统的最优参数. 以下基于图 4-18, 利用对磁盘驱动器系统的参数优化, 其中抗体突变规则改为

$$Ab \leftarrow Ab + r\eta,$$

η 为 \mathbb{R}^5 中的随机变量, 其服从均值为 0, 方差为 1 的正态分布或均匀分布, r 为突变半径, 在此 r 可被自适应地调整以及算法 NIA 中克隆选择率可用 T 细胞的免疫调节机制进行动态调节. 参数选择: 终止迭代数为 500, 群体规模为 80, 小生境法的抑制半径为 4, 记忆细胞数为 10, 随机产生的新抗体数为 6. 在采样数为 100 的条件下, 依据采样周期 h 及输入选择的不同, 分别对该算法进行测试. 当 u_c 选择为单调递增函数, 即

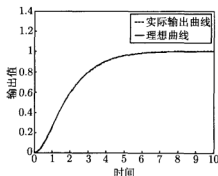
$$u_c(x) = \frac{1}{1 + e^{-x}},$$

以及采样周期分别为 $h=0.1, 0.5, 0.95$ 时, 相应分别获图 4-19(1)~4-19(3), 其中实线表示期望输出的采样曲线, 虚线表示经由算法 NIA 获最优参数所对应磁盘驱动器的磁头臂位置实际输出曲线. 所获最优参数组及最小均方差值由表 4-2 可知.

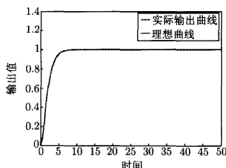
由图 4-2 及表 4-3 可看出, 算法 NIA 能搜索到最优参数组. 采样周期对实际输出与期望输出的吻合程度无多大影响, 但采样周期不适过小, 否则将加大计算复杂度; 无论采样周期大小, 所获实际输出皆较理想. 另外为了进一步检测参考输入

对系统输出的影响,在此引入方波函数作为输入,即

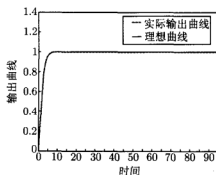
$$u_c(x) = \begin{cases} \frac{1}{2}, & 2k \leq x < 2k+1, \\ -\frac{1}{2}, & 2k+1 \leq x < 2(k+1), \end{cases} \quad k \in \mathbb{N}$$



(1)



(2)



(3)

图 4-19 (1) 采样周期 $h=0.1$; (2) 采样周期 $h=0.5$; (3) 采样周期 $h=0.95$

表 4-2 相同采样个数及不同采样周期的最优参数组和最小均方差值

采样 周期 h	最优参数组					均方差值
	a	b	k	K	J	
0.1	4.13401	0.644558	-1.69956	-1.44482	0.282928	1.35399×10^{-3}
0.5	3.03048	0.644432	-2.35277	-1.0636	0.415968	9.29719×10^{-4}
0.95	3.36428	0.72143	-2.70004	-3.3493	1.39162	3.21063×10^{-4}

当采样周期 $h=0.1$ 时,获得最优参数组 $(a^*, b^*, k^*, K^*, J^*)$ 为 $(-0.018551, -0.187512, -0.036283, -0.826094, 0.25987)$, 均方差值为 0.333219 , 实际输出与期望输出曲线如图 4-20。由该图可知, 尽管输入函数的性质较差, 算法 NIA 也能获得最优参数组, 使得随着时间的延长, 实际输出较贴近期望输出。

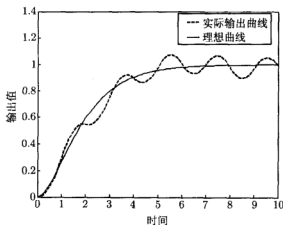


图 4-20 实际输出与期望输出的比较

表 4-3 算法比较结果

变量	最好解			
	COIA	SAPA	GeneAS	Kannan
T	0.8750	0.8125	0.9375	1.125
T_h	0.4375	0.4375	0.5	0.625
R	45.334	40.3239	48.329	58.291
L	140.285	200	112.679	43.690
g_1	5.38×10^{-5}	-0.034324	-0.00475	0.000016
g_2	-0.00501364	-0.052847	-0.038941	-0.068904
g_3	-17.7562	-27.105845	-3652.876838	-21.220104
g	-99.715	-40	-127.321	-196.31
$f(x)$	6090.92	6288.7445	6410.3811	7198.0428

另外,通过对磁盘驱动器控制系统的参数辨识,获得算法 NIA 的另一有趣应用,即其可用于确定定常线性系统的参数,使其达到渐渐稳定.以离散系统(DS)为例,在采样周期为 0.1,采样数为 100,参考输入为 0 的条件下,利用算法 NIA 获得当初始状态 $X_0=(-0.5, 0.5, 0)$ 时,最优参数组为 (3.25753, 0.84254, 1.43182, 1.47764, 0.2349),状态的模为 $\|X(100)\|=3.34428 \times 10^{-6}$;当初始状态 $X_0=(75, 76, 77)$ 时,获最优参数组为 (4.81086, 1.12974, -2.68008, -2.8075, 0.346562),状态的模为 $\|X(100)\|=1.15738 \times 10^{-5}$. 由此表明,对于任意初始状态 X_0 ,算法 NIA 皆能获得最优参数组,使得 $\lim_{t \rightarrow \infty} \|X(t)\| = 0$,即系统(DS)是大范围渐渐稳定.

经由算法 NIA 对磁盘驱动器系统的应用的详细描述,获得了该算法对实际问题的应用思路.算法 NIA 中的记忆细胞数目及抗体进化数目和随机产生的新抗体数目相互协调,以及突变操作中突变半径的自适应调整,体现了该算法的自适应能力及并行搜索的特点,特别对于高维优化问题具有搜索速度快,优化结果很理想的优点,因而是一种有效的新智能优化算法.

例 4.7.2 极点配置问题

对于单输入极点配置问题,在系统完全可控条件下,通过计算一变换矩阵的可逆阵便可获增益矩阵,然而对于多输入极点配置问题,传统的方法需要多次重复尝试求解矩阵方程或判别状态矩阵是否为循环矩阵,从算法的角度,这是一种盲目搜索算法,不便于提高计算效率.然而免疫算法在优化问题的解决中表现出较强优越性,为此构建目标函数,利用动态规模免疫算法(VIA)直接搜索增益矩阵的系数.

(一) 问题的提出

考虑如下线性系统:

$$\dot{X} = AX + Bu,$$

其中 $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, $X \in \mathbb{R}^n$, $u \in \mathbb{R}^p$. 极点配置问题是寻找增益矩阵 $K \in \mathbb{R}^{p \times n}$, 使得在状态反馈控制律 $u = -KX + v$ 的条件下, 状态反馈闭环系统的极点为期望极点

$$\dot{X} = (A - BK)X + Bv$$

$\{\lambda_1^*, \lambda_2^*, \dots, \lambda_n^*\}$, 其中 v 为参考输入. 假定 $A - BK$ 的特征多项式为

$$f(K) = \lambda^n + a_n(K)\lambda^{n-1} + a_{n-1}(K)\lambda^{n-2} + \dots + a_1(K)\lambda + a_0(K),$$

其中 $a_i(K)$ 为关于 K 的元素的特征多项式系数. 于是目标准则函数可选择为均方差函数, 即

$$J(K_1, K_2, \dots, K_p) = \frac{1}{2} \sum_{i=1}^n (a_i(K) - \lambda_i^*)^2,$$

其中 K_i 为 K 的第 i 行向量. 相应的优化问题即可描述为

$$(P) \quad \min_{K_1, K_2, \dots, K_p \in \mathbb{R}^n} J(K_1, K_2, \dots, K_p)$$

(二) 极点配置优化结果

考虑如下多输入系统:

$$\dot{X} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} X + \begin{bmatrix} 1 & 2 \\ -1 & 3 \\ 0 & 1 \end{bmatrix} u.$$

易于验证该系统是完全可控的, 其期望配置的极点设为 $\{-1, 2, 2\}$. 这组期望极点所对应的特征多项式为

$$f(\lambda) = \lambda^3 - 5\lambda^2 + 8\lambda - 4.$$

在算法 VIA 应用于极点配置中, 将增益矩阵 K 中的元素排成的序列作为抗体, 抗原视为进化群体中最好解, 参数选择: 抗体的克隆率为 0.6, 初始抗体群为 80, 抗

体的抑制半径被动态调整, 随机产生的新抗体数为 6. 于是在以上线性系统的基础上, 将 VIA 作用于优化问题 (P), 获得此系统的增益矩阵为

$$K^* = \begin{bmatrix} 0.66884 & 1.67995 & -3.1766 \\ -0.33884 & -0.41234 & 1.92268 \end{bmatrix},$$

矩阵 $A - BK^*$ 的特征多项式为

$$f^*(\lambda) = \lambda^3 - 4.99926\lambda^2 + 7.99974\lambda - 3.99949.$$

算法搜索过程中, 当前群体的最小目标值变化如图 4-21 所示. 从算法搜索效果看, 所获结果已趋于理想值; 另外从图 4-21 可知该算法在第 500 代左右, 系统配置的极点与期望配置的极点的均方差已趋于 0, 这说明算法 VIA 在处理高维实际优化问题时也能获得理想的结果.

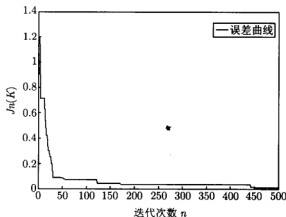


图 4-21 群体最小目标值变化

例 4.7.3 气压钢优化设计

气压钢由圆柱及两个相同的半球帽组成, 设计的目的是使需要的材料、气压钢形状及焊接费用总价最小, 设计所需要的变量是: T_s 圆柱厚度, T_h 球帽厚度, R 内半径, L 圆柱长度, 其数学模型为^[19]

$$\min f(X) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661T_s^2L + 19.84T_s^2R$$

Subject to

$$g_1(X) = -T_s + 0.0193R \leq 0, \quad g_2(X) = -T_h + 0.00954R \leq 0$$

$$g_3(X) = -\pi R^2L - \frac{4}{3}\pi R^3 + 1296000 \leq 0, \quad g_4(X) = L - 240 \leq 0$$

$$0.0625 \leq T_s \leq 6.1875, \quad 1 \leq T_h \leq 6.1875, \quad 10 \leq R \leq 200, \quad 10 \leq L \leq 200,$$

$$X = (T_s, T_h, R, L)$$

在以上模型中, 要求 T_s 及 T_h 是 0.0625 的整数倍. 以下将 COIA 与算法 SAPA^[19]、GeneAS、Kannan 进行比较, 其中 SAPA、GeneAS、Kannan 的比较结果选自文献 [19], 四种算法比较结果如表 4-3.

第八节 本章小结

针对连续空间的优化问题, 为了提高算法寻优速度及效率, 将免疫机理与小生境概念及方法结合获得小生境免疫算法及动态规模免疫算法, 与黄金分割法结合获得约束优化免疫算法, 与模糊逻辑结合获得模糊控制免疫算法, 并对前三种算法进行收敛性论证及性能检测、比较和实际应用, 对模糊控制免疫算法进行性能检测. 理论分析、性能测试、比较及对实际问题的应用表明本章所提出的算法是有效的. 这些算法说明了以免疫系统的简化机制为主、以相关技术为辅设计优化算法解决工程问题具有较大潜力, 特别是约束优化免疫算法, 体现了利用免疫机理开发算法处理约束优化问题的可行性; 其次, 通过模糊规则与免疫机理结合设计模糊控制免疫算法获知, 模糊规则对于设计具有动态特性的优化算法具有较大辅助作用, 这是具有较大潜力的研究课题. 最后为了获知神经网络和免疫算法处理约束优化问题的差异性, 提出一种神经网络, 并进行了理论研究和比较.

参 考 文 献

- [1] Wierchoń S T. Function Optimization by the Immune Metaphor, TASK QUARTERLY2002, 6(3): 1~16
- [2] de Castro L N, Timmis J. An Artificial Immune network for Multimodal Function Optimization. Proc. of Evolutionary Computation (CEC '02), 12-17 May 2002, 1: 699 ~704
- [3] de Castro L N, Von Zuben F J. The Clonal Selection Algorithm with Engineering Applications. In Workshop Proc. of GECC'00, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, USA, July 2000, 36~37
- [4] Yu X, Zheng W, Wu B and Yao X. Solving constrained optimization problems with new penalty function approach using genetic algorithms. Journal of Advanced Computational Intelligence, 1998, 2(6): 208~213
- [5] Rranjithan S R, Chenthan S K, Dakshina H K. Constrained Method-Base Evolutionary Algorithm for Multiobjective Optimization. In Zitzler et al. (eds). Evolutionary Multi-Criteria Optimization, Lecture Notes in Computer Science (LNCS) Springer-Verlag, Berlin, 2001. 1993: 299~313
- [6] Wu B L, Yu X H. Fuzzy Penalty Function Approach for Constrained Function Optimization with Evolutionary Algorithms. Proc. of the 8th International Conference on Neural Information Processing, Fudan University Press, China, 14-18 November 2001, 299~304
- [7] Hajela P, Lee J. Constrained Genetic Search via Schema Adaptation, An Immune Network Solution. In Niels Olhoff and George I. N. Rozvany, editors. Proceedings of the First World Congress of Structural and Multidisciplinary Optimization. Germany: Goslar, 1995. 915~920

- [8] Hajela P, Lee J. Constrained Genetic Search via Schema Adaptation: An Immune Network Solution. *Structural Optimization*, 1996, 12:11~15
- [9] 周明, 孙树栋. 遗传算法及应用. 北京: 国防工业出版社, 2000. 202
- [10] 张著洪, 黄席樾, 胡小兵. 采用重复交叉操作及最优保留策略的遗传算法. *重庆大学学报*, 2002, 25(7): 23~25
- [11] de Castro L N, Von Zuben F J. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transaction on Evolutionary Computation*, 2001, 6(3): 239~251
- [12] 解可新, 韩立兴, 林友联. 最优化方法. 天津: 天津大学出版社, 1997. 321
- [13] 刘海林, 王宇平, 刘永清. 解约束最优化问题的一个新的多目标进化算法. *计算机工程与应用*, 2002, 10, 27~29
- [14] 唐加福, 汪定伟. 模糊优化理论与方法的研究综述. *控制理论与应用*, 2000, 17(2): 159~164
- [15] 区亦勤, 张兴迪. 模糊数学原理及应用. 四川: 成都电讯工程学院出版社, 1989. 474
- [16] Herrera F, Lozano M. Fuzzy Genetic Algorithms: Issues and Models. *Proc. of IPMU'94 (5th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems)*, 1994, 675~680
- [17] 周兆英, 林喜荣, 刘中仁等译. 计算机控制系统 - 原理与设计 (第三版). 北京: 电子工业出版社, 2001. 498
- [18] A.C.C. Carlos. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Computers in Industry* 2000, 41(2): 113~127

第五章 多目标优化免疫算法 及免疫网络算法

第一节 引言

多目标进化算法 (MEA) 已成为解决多目标优化问题的主要工具^[1~3]。经典的优化方法在多目标凸优化方面获得了广泛应用,如权重系数法,此类方法的关键在于确定权重系数,但需大量的先验知识或对初始种群中个体的分布作严格限制。基于群体及 Pareto 概念的 MEA 能很大程度上克服权重系数法的许多缺点,如群体多样性差、先验知识不足及获较少 Pareto 最优解等。这类方法具有较高的并行性,能有效地搜索 Pareto 最优解,其关键在于设计选择操作和适应值方案。在此以文献 [3] 的算法为例加以评述,算法 SPEA^[3] 作为 MEA 领域公认最好的多目标优化算法,充分利用了 Pareto 最优解概念,并用一种新的小生境方法增强种群多样性,同时将所进化的种群中最优个体存于种群外,其参与选择,并通过不断更新所获 Pareto 最优个体。这种方法对于非线性约束凸优化问题较有效,但也存在不少问题,如对于非线性非凸优化问题,该算法较敏感,算法搜索过程中,群体易于沿着单一方向转移,导致获 Pareto 最优解的个数不多,且在各个子目标都同时尽可能达到最小的 Pareto 最优解所在区域上,所获 Pareto 最优解较少,同时当其用于解决各子目标为多模态函数的多目标优化问题时,却很难获得理想结果,不便于决策者作出有效选择。因此对于第二章的第四节中一般非约束多目标优化问题 NCMO 的研究仍处探索之中。为此本章的第三节从体液免疫应答原理出发,利用抗原群和抗体群之间的作用关系提出处理 NCMO 的多目标优化免疫算法,并进行收敛性论证及性能测试和比较。

约束多目标最优化问题 CMO 作为极困难的优化问题,一直是最优化领域关注的热点。许多经典的 MEA 能有效解决凸优化问题,但解决第二章第四节中一般性 CMO 的研究仍然在不断探索之中。尽管算法 SPEA 处理无约束的一般凸多目标优化问题具有很好性能,但实验表明,该方法处理 CMO 却表现出极大缺陷。通常解决 CMO 的方法是权重系数法;如文献 [4] 利用权衡方法 (trade-off method) 将 CMO 转化为单目标约束优化,通过不断调整约束阈值及重复应用 EA,可获多个 Pareto 最优解,但约束阈值的调整依赖于问题本身的 Pareto 最优解集所覆盖的范围,而且此方法对高维问题并非有效(此已在该文中强调)。另外,MEA 处理 CMO 的关键困难在于对初始群体分布有较强的依赖性,在群体分布不均匀时,易出现群体多

样性差. 因此寻求既能自我调节群体多样性, 又能有效解决一般 CMO 的智能新方法已成为计算智能领域研究的主要课题. 目前, 对于免疫遗传算法^[5~7]解决问题 CMO 方面的研究已初步涉及; 对于从免疫学原理出发建立免疫算法解决 CMO 问题方面的研究刚起步, 如文献 [8] 简单地基于克隆选择原理提出算法 MISA, 并可解决非线性的非凸多目标优化问题, 但解决 CMO 却效果不理想 (此由该文结果获知), 原因在于此算法是一种贪婪搜索算法, 进化群体易沿单向转移, 同时该算法未体现 B 细胞抑制与促进机制. 此方法的特点是将克隆选择原理、亲和成熟机理、Pareto 最优解的概念结合, 其本质上是克隆选择算法. 由此可见, 如何充分挖掘免疫系统的内在机制, 提出一种既能合理体现免疫应答过程, 又能有效解决 CMO 的免疫算法仍属于不断探索之中. 为此本章的第四节基于体液免疫应答原理, 通过引入特定的约束处理方法, 以及充分利用抗体群、记忆细胞群和抗原群的作用关系提出算法解决问题 CMO.

另外, 数据分类一直是模式识别研究的重要问题^[11], 许多相关结果都是基于监督学习 (或称为有教师学习) 构建聚类算法, 但由于这些算法对问题的依赖程度较高, 给数据分类产生极大困难. 模糊 C-均值聚类算法^[11] 能克服监督学习分类的许多不足, 但需对已知样本事先确定分类数. 然而, 免疫系统是高度并行的分布式信息处理系统, 受到免疫网络理论的思想启发, 以及免疫系统中抗体群学习抗原具有无监督的特性, 国际上几位学者开发了几种免疫网络算法^[11~21] 用于模式识别中 DNA 识别或数据聚类等, 这些算法在工程领域获得了有效应用, 其共同的特点是皆为非监督学习算法, 并具有突现性质, 因而开发免疫网络算法解决实际问题是一很有发展前景的研究方向. 本章对基于免疫网络的算法的研究初步涉及, 所提出算法的生物基础是独特型免疫网络原理, 算法的作用是能将一组无规则的数组自动分类.

第二节 预备知识

含约束条件 (或无约束条件) 的多目标极小化模型一般可描述为第二章第四节中的问题 CMO (或问题 NCMO). 在这两类问题中, 区域 D 称为有界可行域, \hat{D} (即满足约束的可行解集) 称为约束集. 由于区间 $[0, 1]$ 与区间 $[a, b]$ 之间存在可逆映射, 因此不妨设 D 中元素的每一分量皆在区间 $[0, 1]$ 上取值, 其次不妨设 $f(x) \geq 0$. 对于问题 NCMO, $x \in D$ 称为可行解; 对于问题 CMO, $x \in \hat{D}$ 称为可行解, 若 $x \in D$, 但 $x \notin \hat{D}$, 则称 x 为非法解. 对于 $x = (x_1, x_2, \dots, x_p) \in \mathbb{R}^p$, 约定

$$\begin{aligned} x \geq 0 &\Leftrightarrow x_i \geq 0, \quad 1 \leq i \leq p, \quad \exists i_0, \text{ s.t. } x_{i_0} > 0, \quad 1 \leq i_0 \leq p, \\ x = 0 &\Leftrightarrow x_i = 0, \quad 1 \leq i \leq p \end{aligned}$$

定义 5.2.1^[9] 设 $x_i = (x_{i1}, x_{i2}, \dots, x_{ip}), i = 1, 2$, 若 x_1, x_2 满足 $x_1 \leq x_2$, 则称

x_1 优于 x_2 , 或称 x_2 被 x_1 控制.

定义 5.2.2^[9] 设 $x^* \in D(\text{或 } \hat{D})$, 若在 $D(\text{或 } \hat{D})$ 中不存在 x 使得

$$f(x^*) \leq f(x),$$

则称 x^* 是问题 NCMO(或问题 CMO) 的 Pareto 最优解.

为了以下算法描述方便, 引入

定义 5.2.3 称 $D(\text{或 } \hat{D})$ 的非空有限子集 X 为群体, $x \in D$ 称为个体, 对于问题 CMO, $x \in \hat{D}$ 称为有效个体, $x \in D$ 但 $x \notin \hat{D}$, 则称 x 为非法个体.

定义 5.2.4 设 $x^* \in X \subset D(\text{或 } x^* \in X \subset \hat{D})$. 若在 X 中不存在 $x \in X$, 使得 $f(x)$ 优于 $f(x^*)$, 则称 x^* 是问题 NCMO(或问题 CMO) 相对于群体 X 的 Pareto 最优个体, 简称 Pareto 最优个体. 若 $f(x^*)$ 优于 X 中其他个体的目标向量的数目最多, 则称 x^* 是问题 NCMO(或问题 CMO) 相对于群体的弱 Pareto 最优个体, 简称弱 Pareto 最优个体.

从定义 5.2.4 知, 群体 X 的 Pareto 最优个体必是弱 Pareto 最优个体, 反之则不然. 若 X 为有限集时, 群体 X 的所有弱 Pareto 最优个体皆为 Pareto 最优个体.

第三节 非约束条件下多目标优化免疫算法

在免疫系统与外在环境的作用中, 通常不是 B 细胞群与单个抗原的作用, 而是与多个抗原同时发生作用, 这就形成 B 细胞群与抗原群的作用机制, B 细胞所携带的抗体不仅被抗原群识别, 而且还识别或被 B 细胞群中其他 B 细胞所携带的抗体识别, 因此衡量抗体在免疫系统中的综合能力不仅需考虑抗体应答抗原群的能力而且需考虑被其他抗体激活的能力, 即抗体的激励度. 在此部分, 模拟抗原群与 B 细胞群的相互作用机理, 提出非约束多目标优化免疫算法 (NMOIA: nonconstrained multiobjective optimization immune algorithm) 解决问题 NCMO.

NMOIA 与问题 NCMO 的对应中, 将 B 细胞视为抗体, 而抗体 Ab 被视为问题 NCMO 的可行解, 抗原 Ag 对应问题 NCMO 的 Pareto 最优个体, 即进化群体中的 Pareto 最优个体. 抗体和抗原均用可行解的分量构成的序列表示, 即 $x_1 x_2 \cdots x_p$, 每一分量 x_i 表示为 $0.a_{i1}a_{i2} \cdots a_{i4}$, 每个基因 a_i 取值于 0 到 9 之间的整数, 抗体群及抗原群的规模被动态确定.

一、非约束多目标优化免疫算法原理

NMOIA 是基于体液免疫应答原理, 并通过引入聚类算法处理抗原群中冗余抗原而提出的. 该算法首先设置抗原群集合搜集进化群体中最好抗体 (Pareto 最优个体, 被当作抗原), 并通过聚类消除冗余的抗原, 然后此抗原群对当前进化抗体群的

进化起辅助作用,加速抗体群的进化,但抗原群自身不参与抗体群进化;然后设计免疫算子并模拟抗体群应答抗原群的过程,对进化的抗体群进行进化,最终获得的抗原群即为问题 NCMO 的 Pareto 最优解。此算法由两种类型的算子有机组合而成,即一是抗原处理算子,此搜集抗原及对抗原聚类;二是抗体群应答抗原的免疫算子,即克隆选择、细胞克隆、亲和突变、免疫选择、网络抑制及募集新成员,这些算子中,网络抑制与第三章中克隆抑制的区别是这操作采用抗体间的欧氏距离度量,其他算子均按第三章所出现的相应算子进行定义,但算子的具体设计有所差异,各算子之间相互作用机制如图 5-1 所示。

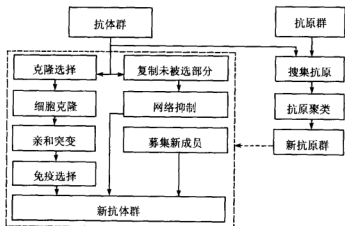


图 5-1 算法 MOIA 的运行机制

在图 5-1 中,虚线框内表示抗体群进化部分,粗箭头符号表示新抗原群作用于当前抗体群,促成抗体群进化。

二、免疫算子设计

符号 S 与 S^N 的含义与前一章描述相同。对于给定的抗原群 Y , 抗原刺激免疫系统,使得抗体群 X 中的部分抗体被激活,抗体 Ab 与抗原群 Y 的亲合力由下确定

$$aff(Ab) = \frac{1}{|Y| \sum_{j=1}^N \langle f(Ab), f(Ag_j) \rangle}, \quad Ab \in X, Ag_j \in Y \quad (5.3.1)$$

其中 $\langle \cdot, \cdot \rangle$ 表示 n 维向量的内积。在此,用内积比用欧氏距离更能确切地刻画抗体应答抗原群的总体能力。式 (5.3.1) 表明,亲合力越高的抗体应答抗原群的能力越强;反之,则越弱。

设 $x, y \in X$, 若 $f(x) \leq f(y)$, 则置 $d_y(x) = 1$, 否则 $d_y(x) = 0$. 于是可引入 X 中抗体的增强度 (即指抗体在抗体群中比其他抗体优越的程度)

$$\text{strength}(x) = \sum_{y \in X} d_y(x). \quad (5.3.2)$$

由定义获知, 当 X 为有限集时, X 中增强度最高的抗体必为 Pareto 最优个体. 以下仅对细胞克隆操作中抗体的克隆数目确定、亲和突变规则作具体说明.

(1) 细胞克隆. 设 X, Y 分别为给定的抗体群及抗原群, 对于 $Ab \in X$, 按其繁殖率 λ 复制 N_{Ab} 个克隆细胞, 即

$$N_{Ab} = \left(|Y| - \frac{1}{\lambda \cdot \text{aff}(Ab)} \right)^{\theta}, \lambda \in \left[\frac{1}{2(1 + \text{aff}(Ab))}, \frac{1}{(1 + \text{aff}(Ab))} \right],$$

其中 λ 为随机数, $1 < \theta < 1.5$ 为参数.

此设计主要体现抗体繁殖的克隆细胞个数与其亲和力成正比, 且每一抗体所繁殖的克隆细胞个数都大于给定抗原群的规模, 其目的在于克隆数在一定范围内, 会增强群体多样性, 但克隆数也不适合过大.

(2) 亲和突变. 设 Ag 为给定的抗原, 对于抗体 Ab 可按如下方式突变:

(a) 超变异: 与第三章动态规模免疫算法的突变规则相同.

(b) 均匀随机突变: 抗体 Ab 以突变率 α_0 作为概率对其各基因位置上的基因在 0 到 9 的整数之间随机突变, 其中 α_0 由式 (5.3.3) 确定. 基因位置指 Ab 的小数点后的数所对应的位置.

$$\alpha_0 = 1 - \lambda \exp(-||Ag - Ab||), 0 < \lambda < 1. \quad (5.3.3)$$

高亲和力抗体所繁殖的克隆经突变后, 有的细胞的亲和力提高, 有的却降低, 而亲和力降低的细胞将被自我抗体或已存在的其他抗体抑制并最终被消除. 为此引入免疫选择算子.

(3) 免疫选择. 将抗体的亲和力取代第三章免疫选择算子中抗体的激励度, 并采用模拟退火选择方案进行选择抗体.

三、抗原聚类算法

为了定量地度量抗原群中抗原之间的关系, 引入二元函数 $d: R \times R \rightarrow R$, 其被给定为

$$d(x, y) = \begin{cases} 0, & x > y, \\ 1, & x \leq y. \end{cases}$$

由于抗原群 Y 中抗原之间具有相互竞争, 这些抗原对免疫系统的激励具有强弱之分, 因此抗原群 Y 中每一抗原有激活其他抗原的能力, 这称为激活力, 其由下式确定

$$D(Ag) = \frac{\sum_{j=1}^{|Y|} R(f(Ag), f(Ag_j))}{\sum_{i,j=1}^{|Y|} R(f(Ag_i), f(Ag_j))}, \quad Ag, Ag_i, Ag_j \in Y,$$

其中

$$R(F_1, F_2) = \sum_{i=1}^m d(x_i, y_i), \quad F_1 = [x_1, x_2, \dots, x_m]^T, \quad F_2 = [y_1, y_2, \dots, y_m]^T$$

于是抗原群聚类算法可描述如下:

Step 1 置初始抗原群为 C .

Step 2 将 C 按

$$\|Ag_i - Ag_j\| < \sigma \quad (5.3.4)$$

划分为子群, 选出每个子群中激活力最高的一个抗原构成抗原群 C' .

Step 3 若 $|C'| > N_0$, 则 $C \leftarrow C'$, 适当调节 σ 并返回 step 2; 否则结束.

四、非约束多目标优化免疫算法描述

利用设计的免疫算子及抗原群聚类算法, 并结合图 5-1, 算法 NMOIA 可描述如下:

Step 1 随机产生规模为 N_0 的初始抗体群 A_0 , 初始抗原群 B_0 为空集.

Step 2 复制 A_n 中增强度最高的抗体作为抗原入抗原群 B_n , 并用抗原聚类算法消去 B_n 中冗余的抗原; 若 $A_n \cap B_n = \emptyset$, 则复制 A_n 中最高增强度的一个抗体作为抗原进入 B_n 中, 此抗原群作用于抗体群 A_n .

Step 3 按选择率 α 将 A_n 作用克隆选择算子, 选择 $N_n \equiv \text{round}(\alpha|A_n|)$ 个亲和力和较高的抗体构成抗体群 A_{n1} , 其余的抗体构成 A_{n2} .

Step 4 将 A_{n2} 作用网络抑制算子, 获记忆细胞群 A_{n21} .

Step 5 将 A_{n1} 作用细胞克隆及亲和突变算子, 每个抗体 Ab 所繁殖的 N_{Ab} 个克隆中 $|B_n|$ 个克隆相应与 B_n 中抗原作用超突变; 对其余的每个克隆 Ab , 随机选择 B_n 中的一个抗原 Ag , 并将 Ab 作用均匀突变; 将此两步所获的突变克隆组合并消除相同的克隆, 获抗体群 A_{n11} .

Step 6 A_{n11} 作用免疫选择算子, 选择 $\text{round}(\eta|A_{n11}|)$ 个抗体构成抗体群 A_{n12} .

Step 7 随机产生 $d_n \equiv \text{round}(\mu|A_n| + 1)$ 个自我抗体插入免疫细胞群 $A_{n21} \cup A_{n12}$, 获 A_{n+1} .

Step 8 若满足终止条件, 则结束; 否则, 返回 Step 2.

此算法有许多特征, 如: (1) 群体规模动态调节及群体并行搜索 Pareto 最优解; (2) 合理体现 Pareto 最优解的概念; (3) 突变概率与亲和力成反比; (4) 随机产生新抗体微调群体多样性、提高搜索速度及维持动态平衡; (5) Pareto 最优个体被保存于抗原群且不断更新. (6) 抗体的克隆数目由其亲和力和抗原群的规模确定.

五、非约束多目标优化免疫算法的收敛性

算法在搜索过程中, 群体规模被动态确定, 并且为有限数, 此可由下结论获知:

定理 5.3.1 若 $0 < \mu < \alpha(1 - \eta N_0^\theta)$, 则有

$$2 \leq |A_n| \leq |A_1| + \frac{1}{\alpha(1 - N_0^\theta \eta) - \mu} \equiv N.$$

证明 记 $a_n = |A_n|$. 由算法描述易知

$$|A_n| \geq 2, \quad |A_{n1}| = \text{round}(\alpha a_n), \quad |A_{n21}| \leq \text{round}((1 - \alpha)a_n),$$

$$|A_{n12}| = \text{round}(\eta|A_{n11}|), \quad a_{n+1} = |A_{n12}| + |A_{n21}| + \text{round}(\mu a_n + 1).$$

又由克隆算子可知, 抗体群 A_{n1} 中每一抗体 Ab 繁殖的克隆细胞个数 $m(Ab)$ 及抗原群的规模满足

$$m(Ab) \leq \left(|B_n| - \frac{1}{(\lambda \cdot \text{aff}(Ab))} \right)^\theta, \quad 1 < \frac{1}{\lambda \cdot \text{aff}(Ab)} < 2, \quad |B_n| \leq N_0 + 1,$$

从而

$$|A_{n11}| \leq \alpha \cdot N_0^\theta \cdot a_n.$$

故有

$$a_{n+1} \leq (\eta \cdot \alpha \cdot N_0^\theta + \mu - \alpha + 1)a_n + 1.$$

进而由假设及此式, 通过归纳获结论成立. 证毕.

此定理表明, 在该算法中, 抗体群的规模始终是有限数, 且由于新抗体产生的随机性, 抗体群的规模不恒同.

为叙述方便, 称 $S \subseteq N$ 中每一元素为状态, 表示为 s_i . 由该算法描述可知, 搜索过程

$$\begin{aligned} A_n &\rightarrow (A_{n1}, A_{n2}) \rightarrow (A_{n1}, A_{n21}) \rightarrow (A_{n11}, A_{n21}) \\ &\rightarrow A_{n12}, A_{n21}) \rightarrow (A_{n12}, A_{n21}, T^{d_n}(S)) \rightarrow A_{n+1} \end{aligned}$$

构成一随机过程. 用 A_n^i 表示 n 时刻抗体群处于状态 s_i , 则 A_n^i 经一步转移为 A_{n+1}^j 时, 状态 s_j 由三部分构成, 即 $s_j = (s_{j1}, s_{j2}, s_{j3})$, 其中

$$\begin{aligned} A_{n+1}^{j1} &= T_{is} \circ T_m \circ T_c \circ T_s(A_n^i) = A_{n12}^{j1}, \\ A_{n+1}^{j2} &= T_{ch}(A_n^i \setminus T_s(A_n^i)), \quad A_{n+1}^{j3} = T^{d_n}(S) \end{aligned}$$

$T^{d_n}(S)$ 表示随机产生的 d_n 个抗体组成的状态. 由于 A_n 在状态 s_i 条件下, $T^{d_n}(S)$ 仅与状态 s_i 有关, A_{n+1} 仅与时刻 n 所处状态及 B_n 中抗原有关, 与 n 时刻以前的任何状态无关, 同时状态空间 $S^{\leq N}$ 为有限集, 因此该算法可描述为有限马尔可夫链. 但由于免疫选择算子 T_{is} 与时间 n 有关, 因而此马氏链是非齐次的. 设 M^* 为优化问题 NCMO 的 Pareto 最优解构成的集合.

引理 5.3.1 设 $s_i, s_j \in S^{\leq N}$, $s_i \cap M^* \neq \phi$, $s_j \cap M^* = \phi$, 则必有

$$P(A_{n11} = s_j | A_n = s_i) = 0.$$

证明 根据亲和力定义, 亲和力最高的抗体必是 Pareto 最优个体; 又由于克隆选择是确定性映射及 Step3 描述可知, 若 $A_n^i \cap M^* \neq \phi$, 则必存在状态 $s_{k0} \subset s_i$, $s_{k0} \cap M^* \neq \phi$, 使得

$$P(T_s(A_n^i) = s_{k0}) = 1.$$

由 K-C 方程获

$$\begin{aligned} P(A_{n11} = s_j | A_n = s_i) &= P(T_m \circ T_c \circ T_s(A_n^i) = s_j | A_{n11} = s_j) \\ &= \sum_{s_k, s_l \in S^{\leq N}} P(T_s(A_n^i) = s_k) P(T_c(s_k) = s_l) P(T_m(s_l) = s_j | A_{n11} = s_j) \\ &= \sum_{s_l \subset s_{k0}} P(T_c(s_{k0}) = s_l) P(T_m(s_l) = s_j | A_{n11} = s_j), \end{aligned} \quad (5.3.5)$$

又由该算法的 Step2 描述及超突变得

$$P(T_m(s_l) \cap M^* = \phi | s_l \cap M^* \neq \phi) = 0. \quad (5.3.6)$$

于是, 当 $s_i \cap M^* \neq \phi$, $s_j \cap M^* = \phi$ 时, 由式 (5.3.5) 及式 (5.3.6) 可获该命题成立. 证毕.

引理 5.3.2 设 $s_i, s_{j1} \in S^{\leq N}$, $s_i \cap M^* \neq \phi$, $s_{j1} \cap M^* = \phi$, 则

$$P(A_{n+1}^{j1} | A_n^i) \leq \varepsilon_n, \quad \varepsilon_n \rightarrow 0, n \rightarrow \infty.$$

证明 由引理 5.3.1 知, 存在状态 $s_{k0} \subset s_i$, $s_{k0} \cap M^* \neq \phi$, 使得对于 $\forall s_k \in S^{\leq N}$, $s_k \cap M^* = \phi$, 必有

$$P(A_{n11} = s_k | A_{n1} = s_{k0}) = 0,$$

进而由 K-C 方程知

$$P(A_{n+1}^{j_1} | A_n^i) = \sum_{s_l \cap M^* \neq \emptyset, s_{j_1} \subset s_l} P(T_m \circ T_c(s_{k_0}) = s_l | A_{n1} = s_{k_0}, A_{n11} = s_l), \quad (5.3.7)$$

$$P(T_{is}(s_l) = s_{j_1} | A_{n11} = s_l).$$

若 $s_l \cap M^* \neq \emptyset, s_{j_1} \subset s_l$, 则存在 s_l 的 Pareto 最优个体 x^* 及 $x_0 \in s_{j_1}$, 使得 $f(x^*) \leq f(x_0)$, 从而

$$\begin{aligned} P(T_{is}(s_l) = s_{j_1} | A_{n11} = s_l) &\leq \prod_{x \in s_{j_1}} \frac{\exp(\text{aff}(x)/T_n)}{\sum_{y \in s_0} \exp(\text{aff}(y)/T_n)} \\ &\leq \frac{\exp(\text{aff}(x_0)/T_n)}{\exp(\text{aff}(x^*)/T_n)} \\ &= \exp\left(-\frac{\text{aff}(x^*) - \text{aff}(x_0)}{T_n}\right). \end{aligned} \quad (5.3.8)$$

又由于

$$\sum_{j=1}^{|B_n|} \langle f(x), f(Ag_j) \rangle \leq m_f^2 |B_n| \leq m_f^2 (N_0 + 1), \quad (5.3.9)$$

$$\sum_{j=1}^{|B_n|} \langle f(x_0) - f(x^*), f(Ag_j) \rangle > 0, \quad (5.3.10)$$

其中

$$m_f^2 = \max_{x \in S} \|f(x)\|,$$

令

$$\rho_0 = \min\{\langle f(x) - f(y), f(y_v) \rangle : f(x) \geq f(y), x, y, y_v \in S\}, \quad (5.3.11)$$

则结合式 (5.3.9)~(5.3.11) 获知

$$\begin{aligned} \text{aff}(x^*) - \text{aff}(x) &= \frac{\sum_{j=1}^{|B_n|} \langle f(x) - f(x^*), f(Ag_j) \rangle}{\sum_{j=1}^{|B_n|} \langle f(x), f(Ag_j) \rangle \sum_{j=1}^{|B_n|} \langle f(x^*), f(Ag_j) \rangle} \geq \frac{\rho_0}{(m_f^2 (N_0 + 1))^2}. \end{aligned} \quad (5.3.12)$$

由于 S 是有限集, 则 $\rho_0 > 0$, 进而结合式 (5.3.7)、(5.3.8)、(5.3.12) 得

$$P(A_{n+1}^{j_1} | A_n^i) \leq |S^N| \exp\left(-\frac{\rho_0}{(m_f^2 (N_0 + 1))^2 T_n}\right) \equiv \varepsilon_n.$$

证毕.

引理 5.3.3 对于群体序列 $\{A_n, n \geq 0\}$, 必有

$$P(A_{n+1} \cap M^* = \phi | A_n \cap M^* = \phi) \leq \left(1 - \frac{|M^*|}{|S|}\right)^{1+\mu}.$$

证明 在条件 $A_n = s_i$ 下, 必有

$$\begin{aligned} & P(A_{n+1} \cap M^* \neq \phi | A_n \cap M^* = \phi) \\ &= \sum_{s_i \cap M^* = \phi, s_j \cap M^* \neq \phi} P(A_{n+1} = s_j | A_n = s_i) \geq P(A_{n+1}^{j_3} \cap M^* \neq \phi | A_n = s_i). \end{aligned} \quad (5.3.13)$$

由于 $A_{n+1}^{j_3} \cap M^* \neq \phi$ 仅与状态 s_i 的群体规模有关, 而与 s_i 中的元素无关, 且

$$P(A_{n+1}^{j_3}) = \prod_{m=1}^{|j_3|} (T(S) = j_{3m}),$$

其中 j_{3i} 表示 s_{j_3} 中的元素, 从而

$$P(A_{n+1}^{j_3} \cap M^* \neq \phi | A_n = s_i) = 1 - \prod_{m=1}^{|s_{j_3}|} P(T(S) = j_{3m}, j_{3m} \notin M^*). \quad (5.3.14)$$

而

$$P(T(S) = j_{3m}, j_{3m} \in M^*) = \frac{|M^*|}{|S|}, \quad (5.3.15)$$

因此由式 (5.3.14)、(5.3.15), 有

$$P(A_{n+1}^{j_3} \cap M^* \neq \phi | A_n^i) = 1 - \left(1 - \frac{|M^*|}{|S|}\right)^{|s_{j_3}|}.$$

又由于

$$|s_{j_3}| = \text{round}(\mu|s_i| + 1) \geq 1 + \mu,$$

故

$$P(A_{n+1}^{j_3} \cap M^* \neq \phi | A_n = s_i) \geq 1 - \left(1 - \frac{|M^*|}{|S|}\right)^{1+\mu}, \quad (5.3.16)$$

从而由式 (5.3.13) 及式 (5.3.16) 获结论成立. 证毕.

定理 5.3.2 若 $0 < \mu < \xi(1 - \eta N^\theta)$, 则多目标优化免疫 NMOIA 算法是概率弱收敛.

证明 由全概率公式得

$$\begin{aligned} P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) &= \sum_{s_i \cap M^* \neq \phi} \sum_{s_j \cap M^* = \phi} P(A_{n+1} = s_j | A_n = s_i), \\ P(A_{n+1} = s_j | A_n = s_i) &= P(A_{n+1}^{j_1} | A_n = s_i) P(A_{n+1}^{j_2} | A_n = s_i) P(A_{n+1}^{j_3} | A_n = s_i) \\ &\leq P(A_{n+1}^{j_1} | A_n = s_i), \end{aligned}$$

从而由引理 5.3.2 可获

$$P(A_{n+1} \cap M^* = \phi | A_n \cap M^* \neq \phi) \leq (2^{|S|} - 1)^2 \varepsilon_n \equiv \beta_n.$$

另外, 由引理 5.3.3 可知

$$P(A_{n+1} \cap M^* = \phi | A_n \cap M^* = \phi) \leq \left(1 - \frac{|M^*|}{|S|}\right)^{1+\mu} \equiv \delta,$$

故由定理 3.6.1 获该定理成立. 证毕.

六、多目标优化免疫算法仿真比较

为了验证 NMOIA 解决问题 NCMO 的有效性, 将此算法与文献 [3] 中算法 SPEA 进行比较. 在比较中, 迭代次数为 200, 对于 SPEA, 参数选定为: 群体大小 80, 交叉概率 0.6, 突变概率 0.001, 外在集合规模 N_0 为 200; 对于 NMOIA, 选择初始抗体群体为 80, $\alpha = 0.6$, $\eta = 0.8$, $\mu = 0.08$ 抗原群规模 N_0 为 200. 所测试的问题 (P) 如下:

$$(P) \quad \begin{aligned} \min f(x) &= (f_1(x), f_2(x)), \\ \text{s.t. } f_1(x) &= x^2, f_2(x) = (x-2)^2, \quad 0 \leq x \leq 3. \end{aligned}$$

尽管该问题较简单, 但其为通用的测试问题以及该问题的 Pareto 面的性质易于分析, 图形描述如图 5-2.

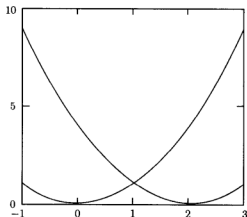


图 5-2 函数 $f_1(x)$ 及 $f_2(x)$ 的图象

以下对 NMOIA 及 SPEA 用于问题 (P) 求 Pareto 最优解. 在此, Pareto 面意指 Pareto 最优解的目标向量构成的曲线; 将 NMOIA 及 SPEA 分别独立作用于问题 (P), 并连续运行 3 次. 每一次独立运行所获 Pareto 面放入同一幅图中, 目的是

为了便于比较,如图 5-3 所示。图中“ ∇ ”表示 MOIA 所获 Pareto 最优解的目标向量,“+”表示 SPEA 所获 Pareto 最优解的目标向量,比较结果见图 5-3。

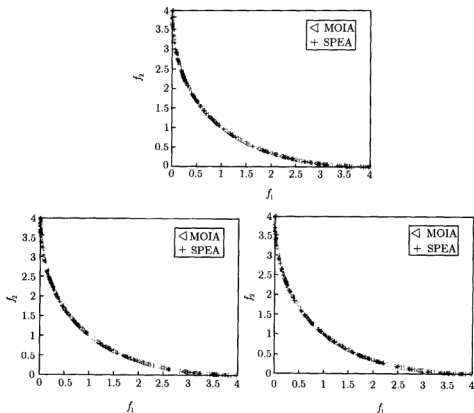


图 5-3 连续三次运行所获 Pareto 面比较

1) NMOIA:113 个, SPEA:94 个; 2) MOIA:114 个, SPEA:87 个; 3) NMOIA:107 个, SPEA:85 个

由图 5-3 获知, NMOIA 比 SPEA 所获 Pareto 最优解个数多,而且显然 NMOIA 比 SPEA 获的 Pareto 面分布好。从这三次连续运行的结果比较可知,第一次 SPEA 所获 Pareto 最优解的分布较差,第二次所获解仅集中在理想 Pareto 面的特定曲线段上,第三次所获解稍微比前两次情形好。这表明 SPEA 处理 NCMO 仍存在严重不足,原因在于 SPEA 对初始群体分布的依赖性。然而, NMOIA 在这三次连续实验中,所获 Pareto 面的分布皆较理想,这说明开发免疫算法处理 NCMO 具有较大潜力。

例 5.3.1 机械铰锤优化设计

机械铰锤设计包含四个变量 d_a, d_b, d_o, l , 如图 5-4 所表示。变量的取值范围、参数取值、图形设计及数学模型均选自文献 [23], 模型描述如下:

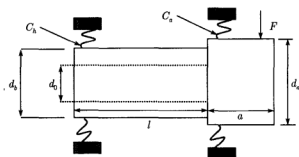


图 5-4 机械铰链设计 [23]

$$\min f(x) = (f_1(x), f_2(x)),$$

$$f_1(x) = \frac{\pi}{4} [a(d_a^2 - d_0^2) + l(d_b^2 - d_0^2)],$$

$$f_2(x) = \frac{Fa^3}{3EI_a} \left(1 + \frac{lI_a}{aI_b}\right) + \frac{F}{c_a} \left[\left(1 + \frac{a}{l}\right)^2 + \frac{c_a a^2}{c_b l^2} \right],$$

$$l_k \leq l \leq l_g, d_{a1} \leq d_a \leq d_{a2}, d_{b1} \leq d_b \leq d_{b2},$$

$$d_0 \leq \frac{d_b}{p_1}, d_b \leq \frac{d_a}{p_2}, \left| \Delta_a + (\Delta_a - \Delta_b) \frac{a}{l} \right| \leq \Delta,$$

$$d_a \in X_1 \equiv \{80, 85, 90, 95\}, d_b \in X_2 \equiv \{75, 80, 85, 90\},$$

其中, p_1, p_2 为正常数, E 为杨氏模量, c_a, c_b 分别是弹簧的承载强度,

$$c_a = 35400 |\delta_{ra}|^{\frac{1}{9}} d_a^{\frac{10}{9}}, c_b = 35400 |\delta_{rb}|^{\frac{1}{9}} d_b^{\frac{10}{9}},$$

δ_{ra}, δ_{rb} 为常数, I_a, I_b 分别惯性力矩,

$$I_a = 0.049(d_a^4 - d_0^4), I_b = 0.049(d_b^4 - d_0^4).$$

其次, 单位不相同及无单位的参数值为: $p_1=1.25, p_2=1.05, E=2.1 \times 10^{-5} \text{ N/mm}^2, F=10^{-5} \text{ N}$; 单位相同的参数值如表 5-1 所示。

表 5-1 参数值表 (单位: mm)

d_{a1}	d_{a2}	d_{b1}	d_{b2}	d_{om}	l_k	l_g
80	95	75	90	25	150	200
a	Δ_a	Δ_b	Δ	δ_{ra}	δ_{rb}	
80	0.0054	-0.0054	0.01	0.001	-0.001	

此问题的主要困难是变量 d_a 及 d_b 均只能取离散值, 解决的方法是在突变概率下, 若 d_a 被突变, 则要求 d_b 也被突变。突变方式有两种, 一是 d_a 及 d_b 分别在 X_1 及 X_2 中变异幅度为 5, 且 d_b 要求满足 $d_b \leq d_a/p_2$; 另一种是 d_a 及 d_b 分别在 X_1 及 X_2 中, 且满足约束不等式。

将算法 NMOIA 用于以上优化问题连续独立运行 4 次, 将这 4 次所去掉非 Pareto 最优解, 最终获得 357 个 Pareto 解, 其图形描述如图 5-5 所示. 由此图易知, 该算法能获得大量 Pareto 解.

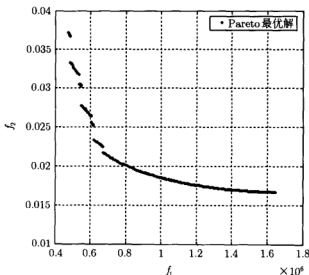


图 5-5 Pareto 面曲线图

综上所述, 算法 NMOIA 是建立在生物免疫理论基础上提出的一种解决 NCMO 的智能优化算法, 此算法的关键在于构建算法框架, 及合理地定义算子模块, 同时该算法充分体现了 Pareto 最优解的概念, 具有群体合作进化和并行搜索等特点.

第四节 约束多目标优化免疫算法

在这一部分, 建立解决约束多目标优化问题 CMO 的免疫算法 (CMIA: constrained multiobjective immune algorithm), 该算法的生物理论基础是克隆选择原理及独特型免疫网络理论, 其模拟了体液免疫应答中免疫细胞的应答、记忆、相互作用等特征, 目的在于充分借助免疫系统的多样性及抗体应答能力, 提出有效的优化算法处理 CMO. 利用克隆选择原理设计抗体群提高识别抗原群的进化框架, 利用独特型免疫网络理论, 建立抗体促进与抑制框架和构建克隆抑制算子, 增强算法搜索过程中群体的多样性, 利用记忆细胞功能保存抗体群中优秀的抗体. 算法设计的一部分机制与第三节中的一些算子的设计类似, 不同的是: (1) 抗原群作用于进化群的方式不同, CMIA 的上一代抗原群作用当前进化抗体群, 而 MOIA 的当前抗原群作用当前抗体群; (2) 利用第四章第四节中的约束条件处理算法确定满足约束的可行解集及对进化过程中出现的非法解作特定处理; (3) 记忆细胞演化算子保存进

化过程中所获得的最好解；(4) 抗体浓度概念被引入增强群体多样性。

一、约束多目标优化免疫算法原理

设抗体对应约束优化问题 CMO 的可行解，自反应细胞被视为非法解，抗原及记忆细胞皆对应问题 CMO 的 Pareto 最优个体。抗体、抗原及记忆细胞皆由 p 个有序的染色体（即问题 CMO 中的决策变量）构成，染色体的长度皆为 l ，所采用的编码为十进制或二进制编码，则所有抗体构成的抗体空间是有限的。CMIA 由 8 种机理构成：克隆选择、细胞克隆、亲和突变、克隆抑制、群体组合、免疫选择、募集新成员及记忆细胞演化，其运行机制如图 5-6 所示。在算法 CMIA 运行之前，首先利用约束处理算法产生抗体集，目的在于回避算法搜索中处理非法解问题，所获得的抗体集用于初始抗体群的产生及修正自反应细胞为抗体，同时也用于募集新成员算子，模拟此运行机制构建算法可解决非线性约束的多目标函数优化（或组合优化）问题。在算法的构建中，引入了亲和力、抗体浓度、增强度及激励度概念，这些概念与第三章及第三节中出现的相应概念的含义相同。抗体激活度指抗体群中抗体被此群体中其他抗体激活的程度，以下对这些概念用数学语言描述。

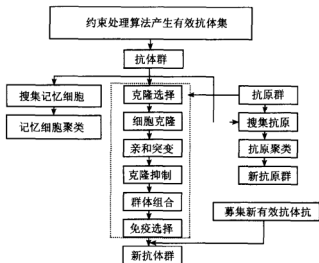


图 5-6 约束优化免疫算法的原理图

设 X, Y 分别表示抗体群及抗原群， X 中抗体 Ab 与抗原群 Y 的亲和力 $aff(Ab)$ 设计为

$$aff(Ab) = \exp \left(- \frac{\sum_{Ag \in Y} \langle f(Ab), f(Ag) \rangle}{\max_{Ab \in X} \sum_{Ag \in Y} \langle f(Ab), f(Ag) \rangle} \right) \quad (5.4.1)$$

设抗体 Ab_i 的 δ 邻域内所含 X 中抗体的数目为 m_i , 此抗体 Ab_i 的浓度被给定为 $c(Ab_i)$ (在此取 δ 为 0.5),

$$c(Ab_i) = \frac{m_i}{|X|}, Ab_i \in X, \quad (5.4.2)$$

X 中抗体 Ab_i 的激活度可定义为

$$F(Ab_i) = \text{strength}(Ab_i) \cdot \exp(-\beta c(Ab_i)), Ab_i \in X. \quad (5.4.3)$$

Ab_i 的激励度可定义为

$$\text{Aff}(Ab_i) = \text{aff}(Ab_i) \cdot \exp(-\beta c(Ab_i)), Ab_i \in X, 0 < \beta < 1, \quad (5.4.4)$$

其中 $\text{strength}(Ab_i)$ 表示抗体 Ab_i 的增强度, 其设计与式 (5.3.2) 相同. 图 5-6 中克隆选择、细胞克隆、亲和突变、免疫选择与第三章的相应算子的定义相同. 以下对部分算子作具体设计:

(1) 细胞克隆算子的设计与上第三节的相应设计相似, 不同的是需用抗体的激励度取代抗体的亲和力.

(2) 亲和突变. 超突变与上第三节的超突变相似, 不同的是, 设 n 为群体进化的代数, 第 n 代抗体群为 X_n , 抗原群为 Y_n , 则克隆的突变率改为

$$\alpha_n = 1 - \exp\left(-\frac{\mu_n + \max_n - \text{aff}(Ab)}{\max_n - \min_n}\right), \mu_n \propto \frac{1}{n}, \quad (5.4.5)$$

$$\max_n(\min_n) = \max(\min)_{Ab \in X_n} \{\text{aff}(Ab)\},$$

其中, μ_n 可选为 $\mu_n = \mu_0 + \frac{T}{n}$, μ_0, T 为调节参数.

(3) 免疫选择. 在给定的抗体群 X 和抗原群 Y 下, 采用第三章免疫选择算子中的模拟退火选择.

(4) 克隆抑制. 在此简单地采用第三章的克隆抑制算子 (二进制编码) 或第五章第三节中的网络抑制算子 (实数编码).

二、约束条件处理及聚类算法

由于问题 CMO 处理较困难, 在此首先采用第四章第四节中约束条件处理算法获得满足约束条件的可行解集, 记为 $Cset$, 此有助于产生 CMOIA 的初始群体及时修改 CMOIA 产生的自反应细胞, 防止退化, 增强进化群体的多样性, 同时也有助于产生自我抗体.

由于抗原群或记忆细胞群中含冗余的个体, 因而引入聚类算法处理群体中过剩的个体, 即聚类算法描述如下:

Step 1 给定群体 $C = \{x_1, x_2, \dots, x_p\}$, 若 $|C| > N_0$ (若 C 为抗原群, 则置 $N_0 \equiv N_1$; 若 C 为记忆细群, 则置 $N_0 \equiv N_2$), 则进入 step2, 否则, 则结束.

Step 2 置 $n = 1$.

Step 3 置 $M = \phi$, 计算 $\|f(x_i)\|$, 依据

$$|\|f(x_i)\| - \|f(x_j)\|| < \sigma + \frac{n-1}{n}, x_i, x_j \in C$$

将 C 划分为 q 个子群 M_k , 不妨设 $p_k = |M_k|$,

$$M_k = \{x_{k1}, x_{k2}, \dots, x_{kp_k}\}, k = 1, 2, \dots, q.$$

Step 4 置 $k = 1$,

Step 4.1 将 M_k 视为抗体群, 利用式 (5.4.3), 计算 $x_{ki} \in M_k$ 的激活度 $F(x_{ki})$.

Step 4.2 根据

$$|F(x_{ki}) - F(x_{kj})| < \sigma_0 + \frac{n-1}{n}, \sigma_0 < \sigma, x_{ki}, x_{kj} \in M_k$$

处罚 x_{ki} 与 x_{kj} 中激活度低的个体, 将未被处罚的个体存入 M 中.

Step 4.3 若 $k < q$, 则置 $k \leftarrow k + 1$, 并返回步 4.1, 否则, 则进入 Step 5.

Step 5 $C \leftarrow M$. 若 $|C| > N_0$, 则置 $n \leftarrow n + 1$, 返回 Step 3; 否则, 结束.

以上算法设计依据是, 个体群 (即抗体群、抗原群、记忆细胞群) 中个体相互激励和抑制的机理, 以及进化论中小生境的思想. 个体的增强度刻画了个体在群体中的优越程度 (通过比较个体间的目标向量体现), 而浓度则刻画了群体中相似个体所占比重. 该算法的特点是个体的激活度反映了个体的浓度和其增强度的关系, 使得在鼓励高增强度个体的同时, 抑制高浓度的个体, 进而增强了群体自我调节能力.

三、约束多目标优化免疫算法的描述

基于 CMOIA 的运行机制、免疫算子设计、约束条件处理算法及聚类算法, CMOIA 可描述如下:

Step 1 若优化问题 CMO 为已处理过的问题或相似问题, 则在记忆集合 M_0 中分别产生 n_0 及 m_0 个个体分别构成初始抗体群和抗原群; 否则, 则分别在 $Cset$ 中随机产生 n_0 及 m_0 个可行解构成初始抗体群体 A_0 及初始抗原群 G_0 , 记忆集合 M_0 为空集.

Step 2 计算抗体群 A_n 中抗体的增强度, 复制 A_n 中 Pareto 最优个体并视为抗原与抗原群 G_{n-1} 组合, 用聚类算法获抗原群 G_n ; 复制 A_n 中 Pareto 最优个体, 并作为记忆细胞入记忆细胞群 M_n , 消除 M_n 中非 Pareto 最优个体, 并用聚类算法获 M_n .

Step 3 计算 A_n 中抗体与抗原群 G_{n-1} 的亲合力; 在选择率 α 下, 选择 A_n 中 $N_n = \text{round}(\alpha|A_n|)$ 个较高亲和力的抗体构成 B_n , 不妨设 $B_n = \{Ab_{n1}, Ab_{n2}, \dots, Ab_{nN_n}\}$.

Step 4 B_n 中 Ab_{ni} 经细胞克隆算子获克隆细胞子群 $C_{ni}, i = 1, 2, \dots, N$, 所有子群构成克隆群体 $C_n, C_n = \{C_{n1}, C_{n2}, \dots, C_{nN_n}\}$.

Step 5 亲和突变及约束处理. 置 $i = 1$.

Step 5.1 在 G_{n-1} 及突变率表达式 (5.4.5) 作用下, 若 C_{ni} 中的克隆为二进制编码的字符串, 则对 C_{ni} 中克隆细胞作用单点突变算子, 获群体 C_{ni}^1 ; 若 C_{ni} 中克隆为十进制编码的数组, 则对 C_{ni} 中 $|G_{n-1}|$ 个克隆依次与 G_{n-1} 中抗原作用超突变算子, 其余 $N_{Ab} - |G_{n-1}|$ 个克隆的染色体分别按突变概率进行均匀突变, 获群体 C_{ni}^1 ; 若突变中出现自反应细胞, 则随机选取 C_{set} 中个体作为抗体取代 C_{ni}^1 中自反应细胞, 若选取的抗体优于父代抗体, 则保留, 否则被父代抗体取代, 进而获临时记忆集 C_{ni}^* .

Step 5.2 若 $i < N_n$, 则 $i \leftarrow i + 1$, 并返回 Step 5.1; 否则进入 Step 6.

Step 6 在 G_{n-1} 作用下, 计算 D_n 中抗体的亲合力,

$$D_n \equiv \bigcup_{i=1}^{N_n} C_{ni}^*,$$

并依据

$$\|Ab_i - Ab_j\| < \sigma, i \neq j, \quad Ab_i, Ab_j \in D_n$$

处罚 Ab_i 和 Ab_j 中亲和力低的抗体, 选择未被处罚的抗体与 A_n 中最优个体构成群体 E_n .

Step 7 在 G_{n-1} 作用下, 计算 E_n 中抗体的激励度, 并用免疫选择算子选择 $|E_n|$ 个抗体构成群体 F_n .

Step 8 在 C_{set} 中随机抽取 d_n 个可行解, $d_n \equiv \text{round}(\mu|A_n| + 1)$, 这些可行解作为抗体取代 F_n 中低亲和力抗体, 获抗体群 A_{n+1} .

Step 9 终止条件判断. 若满足终止条件, 则输出记忆细胞集合中记忆细胞, 否则则返回 Step 2.

该算法通过模拟免疫记忆的机理, 不断更新记忆细胞群, 保留当前抗体群中的优良抗体, 最终获得最佳记忆细胞群体 (即 Pareto 最优解集). Step 3 至 Step 5 模拟了克隆选择原理及亲和成熟机理. Step 7 依据抗体的激励度进行免疫选择, 使得所获群体既含较高亲和力抗体又有一定数量低亲和力抗体, 确保群体的多样性. Step 8 的目的是防止进化群体陷入局部搜索, 增强群体的散布性, 有助于改善搜索性能.

四、约束多目标优化免疫算法仿真比较

(一) 测试问题

为了验证 CMOIA 解决 CMO 的有效性, 将此算法与文献 [3] 的 SPEA 及文献 [8] 的 MISA 进行比较. 指定此三种算法的初始群体规模为 80 (SPEA 及 MISA 的进化群体规模不变), 终止代数 550, 这三种算法的记忆细胞演化操作皆按算法 CMOIA 的方法设计, 记忆细胞集的规模 $N_1 = 80$, 对于下列测试问题 2, 选择 $N_1 = 800$, 其余各参数选定为使各算法对测试问题获较好效果时的参数, 即 SPEA 的交叉概率为 0.6, 突变概率为 0.001; MISA 的突变率为 0.8, 每个抗体克隆的个数为 15 (此参数选自文献 [8]); CMOIA 的参数选定为: 抗原群初始规模为 20, $N_2 = 100$, $\alpha = 51$, $\mu = 8$, $\sigma = 0.5$. 以下列问题 1^[8]、问题 2 及问题 3^[10] 对 CMOIA 进行性能测试, 其中问题 2 由文献 [4] 的两公用测试函数及笔者设计的凸函数组成.

问题 1

$$\text{Maximize } f(x, y) = (f_1(x, y), f_2(x, y))$$

subject to

$$f_1(x, y) = -x^2 + y, f_2(x, y) = \frac{1}{2}x + y + 1,$$

$$x, y \geq 0, \quad 0 \geq \frac{1}{6}x + y - \frac{13}{2}, \quad 0 \geq \frac{1}{2}x + y - \frac{15}{2}, \quad 0 \geq 5x + y - 30.$$

问题 2

$$\text{Minimize } f(x, y) = (f_1(x, y), f_2(x, y), f_3(x, y))$$

subject to

$$f_1(x, y) = 0.5 + (\sin^2 \sqrt{x^2 + y^2} - 0.5) / (1 + 0.001(x^2 + y^2))^2,$$

$$f_2(x, y) = \sum_{i=1}^5 i \cos((i+1)x + i) \sum_{j=1}^5 j \cos((j+1)y + j),$$

$$f_3(x, y) = x^2 + y^2, \quad 2x - 3y + 6 \geq 0, \quad 2x + 3y - 6 \leq 0, \quad x^2 - y - 2 \geq 0.$$

问题 3 车床加工问题

$$\text{Maximize } f(x, y) = (f_1(x, y), f_2(x, y))$$

subject to

$$f_1(x, y) = 6000xy, \quad f_2(x, y) = -\frac{1.28 \times 10^7}{x^{3.33}y^{2.22}},$$

$$xy^{0.75} \leq 95.7, \quad x^2y \geq 1600, \quad 21.1 \leq x \leq 263.9, \quad 0.05 \leq y \leq 1,$$

其中, 问题 3 的 $f_1(x, y)$ 表示金属切削率, $f_2(x, y)$ 表示工具寿命周期, 变量 x 为工具切削速度 (cutting speed), y 为工具每转一周的净给量 (feed rate per revolution).

测试内容包括: (1) 所获 Pareto 最优解的个数; (2) Pareto 面上点的分布密度; (3) CMOIA、SPEA 及 MISA 所获解集之间的覆盖情况。

(二) Pareto 面分布评价准则

(1) 密度的数学描述^[8]: 设 $X = \{x_1, x_2, \dots, x_q\}$ 为 Pareto 最优解集合, 则在由这些解的目标向量构成的曲面上, 分布密度则为 Pareto 面上相邻元素之间的分布, 其定义为

$$S \equiv \sqrt{\frac{1}{q-1} \sum_{i=1}^q (\bar{d} - d_i)^2},$$

$$d_i = \min_{1 \leq j \leq q, j \neq i} (|f_1(x_i) - f_1(x_j)| + |f_2(x_i) - f_2(x_j)|), \bar{d} = \frac{\sum_{1 \leq i \leq q} d_i}{q}.$$

(2) 解集覆盖^[2]: 设 X, Y 分别为由 m 维向量构成的集合, 则 X 覆盖 Y 的覆盖值由下式定义:

$$C(X, Y) = \frac{|\{y \in Y; \exists x \in X : x \geq y\}|}{|Y|}.$$

由此定义可知, $C(X, Y) \neq C(Y, X)$, 另外 $C(X, Y) = 1$ 推出 Y 被 X 完全覆盖, $C(X, Y) = 0$ 推出 Y 中无任何点被覆盖。

(三) 仿真结果比较

上三种算法分别对同一测试问题连续独立运行 4 次 (原因在于 MISA 及 SPEA 易出现在终止条件满足时, 仅获非法解), 将各自所获 4 个记忆集组合, 并消除非优良记忆细胞 (即非 Pareto 最优解), 最终获实际 Pareto 面。在图 5-7 中, 符号 “*”、“+”、“△” 分别表示算法 CMIA、SPEA、MISA 所获的 Pareto 最优解。

问题 1 是一种非凸优化问题, 其理想 Pareto 最优解集中在约束集的较窄范围, 三种算法所获比较值如表 5-2。

由图 5-7(a) 及表 5-2 可知, 从三种算法所获解的数目, 解的散布范围、解的分布及解集的覆盖状况看, CMOIA 优越 MISA, CMOIA 所获 Pareto 面基本拟合理想的 Pareto 面; 尽管 SPEA 所获 Pareto 面的分布好, 但其集中在非常狭窄的范围; 总体上, SPEA 均比 CMOIA 及 MISA 差。

问题 2 是由多峰值函数及凸函数构成的约束优化问题, 所获此问题的 Pareto 面是曲面图, 如图 5-7(b)。三种算法所获比较值如表 5-3。

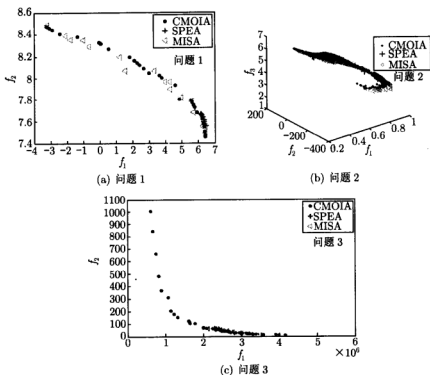


图 5-7 CMOIA、SPEA 及 MISA 用于以上三种测试问题所获实验结果比较

表 5-2 对测试问题 1 的比较

覆盖值	CMOIA	SPEA	MISA	密度	个数
CMOIA	0	0	0.44444	0.26441	41
SPEA	0.26829	0	0.16666	0.037326	6
MISA	0.21951	0	0	0.416427	18

表 5-3 对测试问题 2 的比较

覆盖值	CMOIA	SPEA	MISA	密度	个数
CMOIA	0	0.98156	0.98432	0.1068	713
SPEA	0.00125	0	0.36428	0.1480	248
MISA	0.01458	0.6643	0	0.28283	108

由图 5-7(b) 及表 5-3 可知, MISA 所获解与理想 Pareto 面较贴近, 且分布较均匀; SPEA 所获解的散布性非常差, 而且这些解分别集中在 Pareto 面上的两块较狭窄的区域; CMOIA 所获解的数目最多, 且所获解覆盖面较宽, 这些解与 Pareto 面拟合较好, 从解的分布及覆盖状况看, CMOIA 均优越于 MISA 及 SPEA 所获结果。另外图 5-7(b) 也体现了基于免疫机制的算法在解决多峰值函数优化问题方

面具有较强的群体多样性。SPEA 由于全局搜索能力强而局部搜索能力弱导致所获解的覆盖范围较狭窄。

问题 3 属于实际的非凸优化问题,其难度在于约束集包含有接近坐标轴位置的点,三种算法所获比较值如表 5-4。

表 5-4 对测试问题 3 的比较

覆盖值	CMOIA	SPEA	MISA	密度	个数
CMOIA	0	0.458333	0.214286	0.528072	53
SPEA	0.395833	0	0.178571	0.536124	13
MISA	0.1875	0.258333	0	0.924731	31

由图 5-7(c) 及表 5-4 可知, CMOIA 所获最优解的数目较多,解的分布较好,覆盖的范围较宽,其所获的部分解与 MISA 及 SPEA 所获的解重合,这表明 CMOIA 优于 MISA 及 SPEA; 另外, SPEA 所获解集中在较狭窄范围, MISA 优于 SPEA。

从以上分析获知, SPEA 所获最优解集中在较窄范围,原因在于此算法的多样性差,自适应能力弱等问题。CMOIA 处理复杂的约束优化问题具有较大优势,其所获最优解的分布较均匀,这些解覆盖的范围较宽。事例仿真说明 CMOIA 优越于 SPEA 及 MISA。

值得一提的是,从图 5-7 可看出,算法 MISA 用于以上三种测试问题所获实际 Pareto 面的分布、覆盖范围、与理想解的逼近程度等效果不理想,特别该算法处理问题 2 及问题 3 时,其搜索过程明显表现出群体多样不足,原因在于算法搜索过程中,群体多样性极大依附于该算法的细胞突变算子;在此基础上, Carlos A C C 等对算法 MISA 作了改进,改进后的算法(不妨仍记为 MISA)极大地提高了搜索性能。为了比较改进后的算法 MISA 与算法 CMOIA 的搜索性能,将 CMOIA 中的抑制半径适当减小。由于 CMOIA 的群体规模具有动态调节特点,记忆细胞数依赖于群体规模的变化,这种作用机制更能合理反映免疫系统的动态行为特性,同时实验表明, CMOIA 解决 CMO 可获得较多 Pareto 最优解。现将 CMOIA 作用于以上所述的问题 1,在迭代数 8000 下,获得图 5-8。从此图可看出, CMOIA 对于问题 1 能搜索较多 Pareto 最优解,而且所获解的分布较宽,以及解的分布较均匀。另外图 5-9 描述了改进后的 MISA 用于问题 1 所获得的结果(原始图),在此图中,实线表示理想的 Pareto 面,“+”表示算法所获得的 Pareto 最优解的目标向量。

将图 5-8 与图 5-9 进行比较便可获知,在 $-2 \leq f_1 \leq 8$ 范围内, CMOIA 所获实际 Pareto 面的分布比算法 MISA 的结果好, MISA 所获得的 Pareto 面分布较稀疏,而且在 $-2.5 \leq f_1 \leq -1.5$ 范围, MISA 所获得的 Pareto 解极少;从获解的数量上看, CMOIA 获得 Pareto 解数目极大超过 MISA 获得的解数目;另外从解的覆盖范围看,图 5-8 在 $-8 \leq f_1 \leq -3$ 范围内有多个 Pareto 最优解,在 $-12 \leq f_1 \leq -8$ 范围有一个 Pareto 最优解,而图 5-9 在 $-8 \leq f_1 \leq -3$ 范围内仅获得一个 Pareto

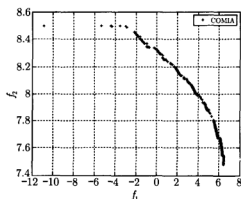


图 5-8 CMOIA 用于问题 1 获实际
Pareto 面

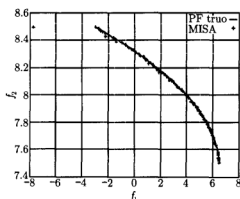


图 5-9 改进后 MISA 用于问题 1 所获
结果原始图

最优解, 在 $-12 \leq f_1 \leq -8$ 范围无 Pareto 最优解, 这表明 CMOIA 所获解集覆盖范围比 MISA 情形要宽。

由此可见, 仅从克隆选择原理出发开发算法解决约束优化问题不能有效地获得期望效果。CMOIA 是从群体相互作用的角度, 将克隆选择原理、亲和成熟机理与独特型免疫网络原理相结合提出的智能算法, 此算法合理地体现了免疫系统与外在环境发生作用的行为机制, 同时通过对测试问题的应用, 获得此算法在搜索性能上优于算法 MISA, 这也表明这算法具有一定应用前景。

综上所述, 算法 CMOIA 是基于生物免疫应答机制并用于解决约束多目标优化问题的一种新方法。该算法的关键在于如何模拟抗体学习抗原的机理, 并合理构建算子模块, 以及如何处理优化问题的约束条件, 结合优化问题的特点, 提出了新的聚类算法。该方法是非线性动态模型, 具有自我调节群体多样性和自适应环境的能力; 并行处理能力强。仿真事例也表明该算法解决优化问题能获得满意效果, 特别适合于处理大系统优化问题。

第五节 模糊免疫网络分类算法

免疫系统作为一种复杂的动力学系统, 其内在的抗体具有识别抗原的作用, 同时抗体之间形成识别与被识别的关系, 像这种抗体与抗体、抗体与抗原的作用关系便构成了免疫网络, 模拟这种网络结构及运行机制可提出免疫网络算法对已知样本进行非监督学习 (或称为无教师学习), 并能动态地确定已知样本组的聚类数及聚类中心。在这一部分, 受到文献 [15] 设计思想的启发, 基于独特型免疫网络理论及模糊逻辑提出一种结构简单且易于应用的模糊免疫网络分类算法, 其由两部分构成, 首先利用独特型免疫网络理论, 构建免疫网络算法, 并用于对已知样本组确定

聚类数和聚类中心；其次，利用其所获结果及聚类损失函数和模糊规则获模糊聚类算法，并对样本逐一归类。

一、免疫网络算法原理

为了寻求既能克服计算复杂度高、特征信息依赖性强，又能克服动态确定聚类数和聚类中心导致误差较大等不足，在此从独特型免疫网络理论出发，建立免疫网络算法；该算法由克隆选择、亲和突变、抑制、抗体组合和吸收新成员等算子组成，这些算子的作用机制由图 5-10 描述。现设抗体 Ab 和抗原 Ag 表示为 p 元数组，抗原代表已知的数据样本，抗体为免疫网络算法所进化的个体，最终所获的个体即为样本组的聚类中心。抗体 Ab 和抗原 Ag 之间的亲和力为 Ab 和 Ag 之间的欧氏距离的倒数。

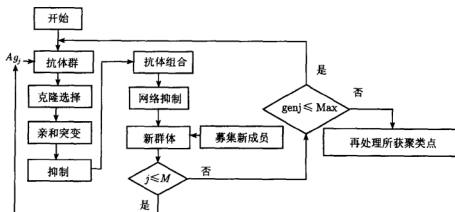


图 5-10 免疫网络算法原理图

二、免疫网络算法的描述

借助于图 5-10 中各算子的运行机制，免疫网络算法的详细步骤描述如下：

Step 1 样本输入。输入 M 个抗原， $Ag_j, j = 1, 2, \dots, M$ 。

Step 2 产生初始抗体群。确定进化代数 $t \leftarrow 1$ ，随机产生初始抗体群 $P(t)$ 。

Step 3 置 $j = 1, Q(j) = P(t), Ag_j$ 作用于进化抗体群 $Q(j)$ 。

Step 4

Step 4.1 克隆选择。选择 $Q(j)$ 中 $\xi\%$ 较高亲和力的抗体构成抗体群 $Q_1(j)$ 。

Step 4.2 亲和突变。对 $Q_1(j)$ 中每一抗体 Ab ，随机产生 $[0, \alpha]$ 上的随机数 β ，使得 Ab 按下式转变为 Ab' ，获抗体群 $Q_2(j)$

$$Ab' \leftarrow Ab + \beta(Ag_j - Ab)$$

其中, α 与 Ab 对 Ag 的亲合力成反比. 此处选定为

$$\alpha = 1 - \lambda \exp(-\|Ag_j - Ab\|), 0 < \lambda < 1.$$

Step 4.3 抑制. 消去 $PQ_2(j)$ 中相同抗体, 然后根据

$$\|Ag_j - Ab\| < \sigma_1, Ab \in P_2(t)$$

消去亲和力低的抗体, 获抗体群 $Q_3(j)$.

Step 4.4 抗体组合. 复制 $Q(j)$ 中 $\eta\%$ 较高亲和力抗体与抗体群 $Q_3(j)$ 中抗体组合构成 $Q_4(j)$.

Step 4.5 网络抑制. 依据下式消去 $Q_4(j)$ 中相同及相似抗体, 获抗体群 $Q_5(j)$,

$$\|Ab_i - Ab_k\| < \sigma_2, Ab_i, Ab_k \in Q_4(t).$$

Step 4.6 依据亲和力大小将抗体群 $Q_5(j)$ 划分为互不相交的子群, 确定各子群的中心, 所有中心构成抗体群 $Q_6(t)$.

Step 4.7 随机产生 $\text{round}(\mu|Q(j)| + 1)$ 个新抗体插入 $Q_6(t)$, 获得群体 $Q(j+1)$.

Step 4.8 内循环判断. 若 $j < M$, 则置 $j \leftarrow j + 1$, 返回 step 4.1; 否则, 进入下一步.

Step 5 外循环判断. 若迭代数 t 小于或等于指定代数, 则置 $t \leftarrow t + 1$, $P(t+1) \leftarrow Q(M)$, 对抗原群随机排序, 返回 step 3; 否则进入下一步.

Step 6 聚类数及类中心的确定. 再次利用步骤 4.6 获聚类数及各类的中心.

以上算法对已知样本采用无教师学习的方式, 自动确定聚类数及聚类中心. step 4.5~4.6 采用了网络抑制的思想; step 4.7 利用了免疫系统中新抗体随机产生的机理. step 1~5 所获的中心点个数一般较多, 有些是多余的, 有些中心点相距较近, 因而应用 step 6 再次处理所获中心点, 从而获已知样本的聚类数及聚类中心.

该算法的特点为: (1) 能并行搜索已知样本组的各类中心及动态确定聚类数及聚类点; (2) 所获聚类中心与实际中心的偏差较小; (3) 对问题无特征信息的依赖.

三、模糊免疫网络分类算法

设 $\{x_i = (x_{i1}, x_{i2}, \dots, x_{ip}) \mid x_{ik} \in [0, 1], i = 1, 2, \dots, n, k = 1, 2, \dots, p\}$ 是 n 个样本组成的样本集, c 为聚类数, 聚类中心集为 $\{M_i = (m_{i1}, m_{i2}, \dots, m_{ip}) \mid m_{ik} \in [0, 1], i = 1, 2, \dots, c, k = 1, 2, \dots, p\}$, 相应的类依次设为 $\hat{A}_1, \hat{A}_2, \dots, \hat{A}_c$. 设 $\mu_j(x_i)$ 表示第 i 个样本对第 j 类 \hat{A}_j 的隶属度, 则由文献 [11] 获聚类损失函数

$$J_f = \sum_{j=1}^c \sum_{i=1}^n (\mu_j(x_i))^b \|x_i - M_j\|^2$$

在条件

$$\sum_{j=1}^c \mu_j(x_i) = 1, i = 1, 2, \dots, n$$

的约束下, 取最小值的必要条件为

$$\mu_j(x_i) = (1/||x_i - M_j||^{2/(b-1)}) / \sum_{k=1}^c ||x_i - M_k||^{2/(b-1)}, i = 1, 2, \dots, n, j = 1, 2, \dots, c, \quad (5.5.1)$$

其中, b 是对聚类的模糊程度度量的常数. 根据式 (5.5.1), 计算样本 x_i 对类 \hat{A}_j 的隶属度, 求出 x_i 对所有类的隶属度中最大隶属度 Max_i , 且设 x_i 对 c_0 个类的隶属度都为 Max_i , 不妨设为 $\hat{A}_1, \hat{A}_2, \dots, \hat{A}_{c_0}$. 若 $c_0 = 1$, 则由最大隶属度原则^[22], x_i 归入 \hat{A}_{c_0} 类; 若 $c_0 > 1$, 则需引入模糊规则将 x_i 归类.

定义 5.5.1 设 A 是论域 U 上的模糊集, x 是论域 U 中的一个元素, 若 x 对 A 的隶属度 $\mu_A(x) > 1/2$, 则称 x 属于模糊集 A 偏真; 若 $\mu_A(x) \leq 1/2$, 则称 x 属于模糊集 A 偏假.

定义 5.5.2 设 $U=[0,1]$ 是一论域, A, B 是 U 上的模糊集, 若隶属度 $\mu_B(\mu_A(x)) > 1/2$, 称 U 上的点 x 相对于模糊集 A 贴近于模糊语句 “ y 属于 B ”.

此外, 设 A_1 表示 “偏小” 的模糊集, A_2 表示 “偏中” 的模糊集, A_3 表示 “偏大” 的模糊集, B 表示 “ x 可归类” 的模糊语句. 于是, 引入隶属度函数—— S 函数^[22], 并置

$$\mu_{A_1}(x) = 1 - S(x; 0, 0.75),$$

$$\mu_{A_2}(x) = \begin{cases} S(x; 0.25, 0.5) & x \leq 0.5 \\ 1 - S(x; 0.5, 0.75) & x > 0.5 \end{cases}, \mu_{A_3}(x) = 1 - S(x; 0, 1).$$

因此, 模糊规则可描述如下:

若 x 属于 A_1 偏真或 x 属于 A_3 偏真且 x 属于 A_2 偏假, 则 y 属于 B . (5.5.2)

当样本 x_i 对 c_0 个类 $\hat{A}_1, \hat{A}_2, \dots, \hat{A}_{c_0}$ 的隶属度都最大时, 定义分量 x_{ik} 对模糊集 A_{i_j} 中所有元素的第 k 个分量构成的模糊集 $A_{i_j}(k)$ 的隶属度为

$$\mu_{A_{i_j}(k)}(x_{ik}) = (x_{ik} - m_{i_j k})^{2/(b-1)} / \sum_{i_j=1}^{c_0} ||x_{ik} - m_{i_j k}||^{2/(b-1)},$$

$$i_j \in \{1, 2, \dots, c_0\}, k = 1, 2, \dots, p.$$

利用定义 5.5.2, 计算 x_i 相对于模糊集 A_{i_j} 贴近模糊规则 (5.5.3) 中模糊条件的贴进度

$$d_{A_{i_j}}(x_i) = \prod_{k=1}^p d'_{A_{i_j}(k)}(x_{ik}), \quad (5.5.3)$$

其中

$$d'_{A_{i_j}(k)}(x_{ik}) = (\mu_{A_1}(\mu_{A_{i_j}(k)}(x_{ik})) \vee \mu_{A_3}(\mu_{A_{i_j}(k)}(x_{ik}))) \wedge (1 - \mu_{A_2}(\mu_{A_{i_j}(k)}(x_{ik}))).$$

选择 c_0 个类 $\hat{A}_1, \hat{A}_2, \dots, \hat{A}_{c_0}$ 中的一个类 A_{i_k} , 使得由式 (5.5.3) 计算 x_i 相对于类 A_{i_k} 贴近于模糊规则 (5.5.2) 中模糊条件的贴近度 $d_{A_{i_k}}(x_i)$ 最大, 从而 x_i 归入类 A_{i_k} . 现描述模糊聚类算法如下:

Step 1 输入 n 个已知样本 $x_i = (x_{i1}, x_{i2}, \dots, x_{ip}), i = 1, 2, \dots, n$.

Step 2 利用免疫网络算法获类的中心 $M_j, j = 1, 2, \dots, c$, 相应的类依次设为 A_1, A_2, \dots, A_c .

Step 3 置 $i = 1$.

Step 4 根据式 (5.5.1) 计算 x_i 对已知的 c 个类的隶属度 $\mu_j(x_i), j = 1, 2, \dots, c$.

Step 5 求出 c 个隶属度值 $\mu_j(x_i), j = 1, 2, \dots, c$, 最大隶属度为 Max_i , 找出使 $\mu_j(x_i) = \text{Max}_i$ 成立的类, 不妨设为 A_1, A_2, \dots, A_{c_0} .

Step 6 若 $c_0=1$, 则 x_i 归入 A_{c_0} 类; 若 $c_0 > 1$, 则进入下一步.

Step 7 根据模糊规则 (5.5.2), 在类 A_1, A_2, \dots, A_{c_0} 中选择一类 A_{i_k} , 使得 x_i 相对于 A_{i_k} 最贴近 (5.5.2) 的模糊条件, 并把 x_i 归入类 A_{i_k} .

Step 8 若 $i \leq n$, 则返回 step 4; 否则, 则结束.

以上算法中, step 3.4 属于同一样本仅对一个类取最大隶属度时, 该样本此时可归类; step 3.6 是针对于同一样本对多个类都取最大隶属度的情形下, 采用模糊规则 (5.5.2) 及式 (5.5.3), 选出最好一类并将样本归入此类, 目的是使分类更准确.

四、实验结果及误差分析

经过对给定的 200 个、随机产生的 2000 个二元数组样本进行聚类, 获知免疫网络聚类算法中的克隆选择参数 ξ , 阈值 d, σ_1, σ_2 和参数 b 对数据分类的质量起重要控制作用, ξ 控制抗体突变个数, d 控制群体的多样性, $\sigma_i (i = 1, 2)$ 确定聚类中心, b 控制模糊聚类结果的模糊程度. 经过多次调试, 获知参数选择范围: $\xi \in (50, 60), \eta \in [80, 90], d \in [5, 8], \sigma_1 \in (0.56, 0.58), \sigma_2 \in (0.7, 0.75), b \in [2.5, 10]$. 以下实验中, 当参数选定为: $\xi = 51, \eta = 85, d = 5.8, \sigma_1 = 0.57, \sigma_2 = 0.71, b = 5$ 时, 分类较理想.

现对给定 74 个二维样本利用免疫网络算法进行非监督学习, 确定聚类数及聚类点, 进而用模糊聚类算法将已知样本归类; 从数据分布来看 (图 5-11 (a) 至图 5-13 (a)), 所给定的数据可大致分为 4 类, 这些数据中仅有一组数据分布较均匀, 其他数据分布不很均匀, 并且有分布较稀疏的数据; 模糊免疫网络分类算法获结果如图 5-11 至图 5-13 所示. 其中 $J[n]$ 表示免疫网络聚类算法的第 n 次迭代所获抗体群视为聚类点的条件下, 利用模糊分类算法获聚类损失函数 J_f 的值.

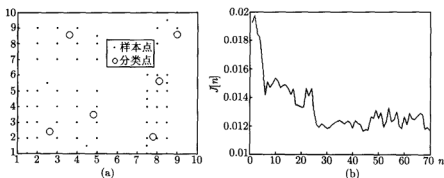


图 5-11 (a)、(b) 分别为第一次聚类点及损失函数变化曲线

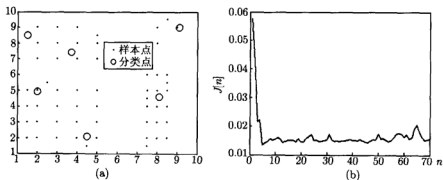


图 5-12 (a)、(b) 分别为第二次聚类点及损失函数的变化曲线

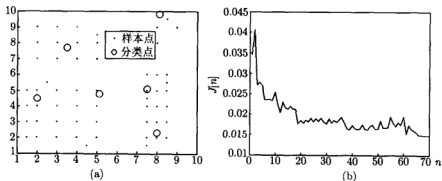


图 5-13 (a)、(b) 分别为第三次聚类点及损失函数的变化曲线

从图 5-11 (a) 至 5-13 (a) 获知, 该算法能将样本分为 6 类, 特别是已知样本中有一组能被分为两类, 这些样本被分为 6 类的效果比较理想; 另外分类损失随着免疫网络进化代数的增加而逐渐减小, 并且当迭代数增加到一定的值后聚类损失便在确定的范围变化, 这说明网络的进化已趋于泛化。

综合以上分析,免疫网络算法与模糊分类算法有机结合能有效进行数据处理,免疫网络算法起到非监督学习样本作用,寻找样本组的聚类点和分类数,而模糊分类算法则准确地完成对样本的分类任务.因此有效借鉴免疫网络原理,合理开发免疫网络算法解决工程领域中关于数据处理问题是值得研究的领域.

第六节 本章小结

基于体液免疫应答的运行机制及记忆细胞功能,从集合的观点,依据抗原群作用免疫系统及抗原相互竞争的思想,将抗体群、抗原群、记忆细胞群紧密结合,并依据其作用关系提出了非约束及约束多目标优化免疫算法.对这两种算法实际应用和比较,以及对非约束多目标优化免疫算法进行收敛性研究.最后从免疫细胞的免疫功能、免疫网络的动态特性出发,以独特型免疫网络理论为支撑,设计了处理数据压缩的免疫网络算法,进而结合模糊逻辑提出模糊免疫网络分类算法处理数据分类问题.

参 考 文 献

- [1] Fonseca C M, Fleming P J. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 1995, 3(1): 1~16
- [2] Zitzler E, Thiele L. Evolution algorithms for multiobjective optimization: methods and application. Doctoral thesis ETH No. 133398, Zurich: Swiss Federal Institute of Technology, 1999. 114
- [3] Zitzler E, Thiele L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 1999, 3(4): 257~271
- [4] Rranjithan S R, Chentana S K, Dakshina H K. Constrained Method-Based Evolutionary Algorithm for Multiobjective Optimization. In Zitzler et al. (eds). *Evolutionary Multi-Criteria Optimization, Lecture Notes in Computer Science (LNCS)*. Berlin: Springer-Verlag, 2001. 1993: 299~313
- [5] Yoo J, Hajela P. Immune network simulations in multicriterion design. *Structural Optimization*, 1999, 18: 85~94
- [6] Yoo J, Hajela P. Fuzzy multicriterion design using immune network simulation. *Struct Multi-disc Optim*, Springer-Verlag, 2001, 22: 188~197
- [7] 孙维学. 基于免疫原理的多目标进化算法群体多样性研究. *模式识别与人工智能*, 2001, 14(3): 291~295
- [8] Carlos A C C and Nareli C C. An Approach to Solve Multiobjective Optimization Problems Based on Artificial Immune System. 1st International Conference on Artificial Immune Systems (ICARIS-2002), University of Kent at Canterbury, 9-11 September 2002, 212~221
- [9] 林桂云, 董加礼. 多目标优化的方法与理论. 吉林: 吉林教育出版社, 1992. 446
- [10] Osyczka A. Multicriterion Optimization in Engineering with FORTRAN Program. Ellis Horwood, John Wiley, New York, Chichester, Brisbane, Toronto 1984. 301
- [11] 边肇祺, 张学工. 模式识别. 第二版. 北京: 清华大学出版社, 2000. 338

- [12] Hunt J E and Cooke D E. An adaptive, distributed learning systems based on the immune system. In Proc. of the IEEE International Conference on Systems, Man and Cybernetics, 1995, 2494~2499
- [13] Hunt J E, Cooke D E. Learning using an artificial immune system. J. of Network and Computer Appl., 1996, 19(4):189~212
- [14] Nasaroui O, Gonzalez F, and Dasgupta D. The Fuzzy Artificial Immune System: Motivations, Basic Concepts, and Application to Clustering and Web Profiling. Proc. of the 2002 IEEE International Conference on Computational Intelligence, 12-17 May 2002, 1: 711~716
- [15] de Castro L N, Von Zuben F J. aiNet: Artificial Immune Network for Data Analysis. In H. A. Abbass H A, Sarker R A, and Newton C S (eds.). Data Mining: A Heuristic Approach. USA: Idea Group Publishing., 2001. 231~259
- [16] de Castro L N, Timmis J. An Artificial Immune network for Multimodal Function Optimization. Evolutionary Computation, Proc. of IEEE CEC '02, 12-17 May 2002, 1: 699 ~704
- [17] Kim D H. Tuning of a PID Controller Using a Artificial Immune Network Model and Local Fuzzy Set. IFSA World Congress and 20th NAFIPS International Conference, 25-28 July 2001, 1.5: 2698~2703
- [18] Kim D H. Parameter Tuning of Fuzzy Neural Networks By Immune Algorithm. Proc. of the 2002 IEEE International Conference on Neural Networks and Computational Intelligence, 12-17 May 2002, 1: 408~413
- [19] Kim D H, Lee K Y. Neural Networks Control by Immune Network Algorithm Based Auto-Weight Function Tuning. Proc. of the 2002 International Joint Conference on Neural Networks and Computational Intelligence, 12-17 May 2002, 2:1469 ~1474
- [20] Timmis J, Neal M, Hunt J. An Artificial Immune System for Data Analysis. Biosystems, 2000, 55(1/3): 143~150
- [21] Timmis J, Neal M, Hunt J. Data Analysis Using Artificial Immune Systems, Cluster Analysis and Kohonen Networks: Some Comparisons. IEEE International Conference (SMC '99) on Systems, Man, and Cybernetics, 12-15, Oct. 1999, 3: 922 ~927
- [22] 区奔勤, 张兴迪. 模糊数学原理及应用. 四川: 成都电讯工程学院出版社, 1989. 474
- [23] Tan K C, Lee T H, Khor E F. Evolutionary Algorithms With Dynamic Population Size and Local Exploration for Multiobjective Optimization. IEEE Transactions on Evolutionary Computation, 2001, 5(64): 565~588

第二部分

图像编码的分形算法

数据压缩的思想起源较早, 汉语中的文言文是典型代表. 在我国古代, 充满智慧的祖先使用言简意赅的文言文表达各种思想. 通常, 短短数字、寥寥数语就能表达十分丰富的思想. 这充分表明, 文言文中字符的信息冗余被降低到了最小程度. 因此, 文言文是消除数据冗余以压缩数据的思想的生动体现.

自 20 世纪 80 年代以来, 计算机在各行各业和社会生活各个方面得到广泛应用, 并已广泛应用于数据处理与数据通讯. 特别是 90 年代以来, 计算机系统的时代特征是应用多媒体技术的发展. 简单地说, 多媒体技术就是指用计算机综合处理文字、声音、图形、图像等多种媒体上承载的信息. 图像可以认为是其中最重要的信息载体——“百闻不如一见”, 这道出了人类感知语言信息与视觉信息的能力的本质差异, 充分反映了信息的图像载体存在很多适合于人类视觉系统的优点. 然而, 即使在压缩格式 (如 JPEG) 下, 图像也需要巨大的存储空间和较长的传输时间. 因此, 尽管多媒体等信息系统涉及方方面面的问题, 每个问题都很重要, 但以较少的空间储存海量文件 (如图像) 的压缩是解决数据有效传输与储存的关键问题. 随着多媒体应用的日益增多, 如何高效、实时地压缩图像是多媒体技术中最关键的问题之一, 图像压缩技术已经成为一个十分重要的研究领域, 发展新的图像压缩技术已成为十分重要的研究课题. 可以预见, 图像压缩技术将会是正在建设的数字化社会所依赖的主要技术基础之一.

从 Oliver 等提出 PCM 编码理论算起, 图像压缩技术已经走过 50 余年的历程, 其间许多如预测编码、变换编码和矢量量化编码等经典压缩编码方法被提出, 并得到较为广泛的实际应用. 众所周知, 经典压缩编码主要依据图像本身固有的统计特性, 较少利用人眼视觉系统的特性, 压缩效率一般要受信息熵的约束, 不能实

现很高的压缩比。然而随着图像技术的广泛应用,特别是通信的实时性对图像压缩比要求越来越高,超高倍的数据压缩对图像通信技术是一个极大的推动,但按经典压缩编码方法是难以实现的。随着感知生理——心理学的发展,人们越来越清楚地认识到视觉感知是一种宏观认识过程,人的视觉感知特点与统计意义上的信息分布有时并不完全一致,统计上需要许多信息量才能表征的某些特征对视觉感知也许并不重要。因此,从感知角度来说,详细表征这部分特征是不必要的。受此启发,人们从微观转向宏观去研究开发新的编码方法,并注重对感知特性的利用,分形编码(fractal coding)等新一代编码技术自然就应运而生。

分形图像编码是美国数学家 Barnsley 在 20 世纪 80 年代末提出的美国专利技术,源于对分形几何组成部分的迭代函数系统(iterated function system)的研究,为图像压缩提供了一条与以往完全不同的新思路。尽管用该技术对几幅图像的分形编码获得了难以置信的超高压缩比(10000:1),但是,这种早期分形编码是不实用的,因为编码时间太长,且需要人机交互,对操作者有较高要求。在分形编码实用化的研究中, Jacquin 迈出了实质性的一步,在其发表于 IEEE 刊物的获奖论文(1992)里提出了一种基于方块划分的计算机自动编码的分形算法——分形块编码(fractal block coding)。自此以后,分形编码成为一个众多学科领域的学者参与的活跃研究领域。

目前,分形编码以其新颖的思想、高压缩比、分辨率无关性和快速解码等优点受到学术界、技术界广泛关注,是目前公认的三种最有前途的新一代图像编码技术之一(另两个是小波技术与模型法技术)。此外,作为分形编码核心基础的分形理论,是非线性科学研究中十分活跃的一个分支,特别是十余年来在计算机图像处理和显示中显示出越来越重要的作用。自然图形的模拟、图像压缩、图像纹理分析、数字水印技术和模式识别等领域的大量学术论文都可见分形编码的应用。

任何新方法提出后,从不同的观点对它加以研究总结总是一件十分有意义的工作。不同的观点一方面能够使我们更好地了解方法的本质,另一方面也能够启发我们进行新的研究。分形编码自然不能例外,事实上,自分形编码出现以来,人们已从多种不同的观点对它进行了深入研究。本书这一部分拟从迭代函数系统的观点介绍分形编码的基本原理,从 VQ 观点介绍其算法及实现。这样做是基于这样的考虑:迭代函数系统是分形编码的起源,是分形编码的传统方法,也是学习分形编码最好的教材。但对于分形压缩软件的设计,VQ 观点更适合、更直接,因为它本身就是直接处理离散对象的。

第六章 分形编码的数学基础

第一节 引言

分形编码的数学基础主要包括迭代函数系统、压缩映象原理和拼贴定理。迭代函数系统 (iterated function system, IFS)^[1] 是 Barnsley 及其研究小组在 Hutchinson 1981 年提出的迭代函数理论 (iterated function theory)^[2] 的基础上发展起来的, 它是分形几何的一个重要组成部分。压缩映象原理是泛函分析中的一个著名结果, 是著名数学家 Banach 在总结前人结果的基础上于 1922 年提炼出来的^[3,4]。拼贴定理^[1] 是 Barnsley 于 20 世纪 80 年代后期提出的, 它是压缩映象原理的一个简单推论, 并成功地用于集合、函数 (包括信号、图像) 等的逼近。

基于 IFS 的分形图像编码可以获得极好的压缩性能^[5,6], 能够实现很高的压缩比^①, 其实质是假设现实图像具有自相似性 (分形的典型特征) 来实现图像数据压缩的。从数学上看, 分形编码的原理是简单的, 待编码图像由不动点接近它的压缩仿射变换表示, 压缩映象原理保证不动点图像由压缩变换迭代作用于任意初始图像来生成, 拼贴定理则保证不动点图像是待编码图像的近似图像。

尽管分形编码的原理从概念的观点上看是容易理解的, 但是为了叙述严谨、清晰, 坚实可靠而明确的数学描述是绝对必要的^[7]。此外, 要了解分形图像压缩技术的起源, 以及理解该技术的数学原理, 分形的概念、度量空间及其压缩映象原理和拼贴定理是必不可少的。此外, 分形理论 (核心是分形几何) 是非线性科学研究中十分活跃的一个分支, 特别是十余年来在计算机图像处理和显示中显示出越来越重要的作用。自然图形的模拟、图像压缩、图像纹理分析、数字水印技术和模式识别等领域的大量学术论文都可见分形的身影。

总之, 分形图像编码是分形几何、泛函分析等现代数学分支最成功的应用之一。这充分说明, 一些复杂的数学理论, 尽管难于理解, 但往往能为图像处理等技术提供意想不到的合理方法。

第二节 度量空间

对于分形图像编码来说, 度量空间的概念是必不可少的, 还要涉及一些集合论、拓扑学、动力系统 etc 数学分支的基本概念。此外, 在分形图像编码框架内, 图

^① The method has yielded compression ratios in excess of 10,000 to 1 (bringing our aerial photo down to a manageable 13,000 bytes)(摘自文献 [6], 遗憾的是该文没有给出方法的具体细节)。

像被看成是 Hilbert 空间的向量^[6], 其度量由内积通过范数诱导出来. 不过, 在短短的篇幅里面面面俱到是不可能的, 也是不必要的. 因此, 我们对这些数学概念作个简短介绍, 许多细节可以参考有关的数学书籍.

一、度 量

极限是分析数学中基本概念之一. 序列 (数列、函数列等) 的许多收敛概念都可以统一在度量空间中按度量 (距离) 收敛的概念之中. 这样的处理, 使我们有可能更容易认识那些乍看似似乎毫无关系的极限过程的本质联系. 此外, 度量概念也可以使不确切的“接近”、“逼近”等说法精确化.

欧氏空间中与拓扑有关的概念, 如开集、闭集等, 能够以自然的方式推广到度量空间中去. 在度量空间中引进相应的概念, 并建立相应理论后, 我们就可以进一步对每个具体空间引出相应的结论.

定义 6.2.1 设 X 是一个非空集合. 函数 $d: X \times X \rightarrow \mathbb{R}$ 称为 X 上的一个度量 (亦称距离), 如果它满足下面的三条度量公理:

(1) 正定性: $d(x, y) \geq 0$, 且 $d(x, y) = 0 \Leftrightarrow x = y, \forall x, y \in X$;

(2) 对称性: $d(x, y) = d(y, x), \forall x, y \in X$;

(3) 三角不等式: $d(x, y) \leq d(x, z) + d(z, y), x, y, z \in X$.

引进了度量 d 的非空集合 X , 称为度量空间, 记为 (X, d) .

顺便指出, 对称性可以由正定性和三角不等式推出. 此外, 正定性中的非负性可以从其余的条件推出. 尽管如此, 在泛函分析文献中, 在定义度量时基本上都同时使用上述三条度量公理.

定义 6.2.2 点列 $\{x_n\} \subset X$ 称为在度量空间 (X, d) 中收敛于 $x \in X$, 如果对于任意 $\varepsilon > 0$, 存在自然数 N , 使得对所有的 $n > N$, 有 $d(x_n, x) < \varepsilon$.

此时, 写作 $\lim_{n \rightarrow \infty} x_n = x$ 或 $x_n \rightarrow x (n \rightarrow \infty)$, 并称 x 是点列 $\{x_n\}$ (按度量 d) 的极限.

现在我们考虑一些阐述上述概念的例子, 但略去验证细节. 它们在分形图像编码中都是十分重要的度量空间.

例 6.2.1 设 $X = \mathbb{R}^n$ 是所有 n 元有序实数组的集合, 对于任意 $x, y \in \mathbb{R}^n$, 定义:

$$d_p(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}, \quad 1 \leq p < \infty, \quad (6.2.1)$$

$$d_\infty(x, y) = \max\{|x_i - y_i| : 1 \leq i \leq n\}, \quad (6.2.2)$$

可以验证它们都是 \mathbb{R}^n 中的度量.

例 6.2.2 设 X 是 n 维 Euclid 空间的可测集 (如区间、矩形等), 定义函数集合:

$$L^p(X) = \left\{ u(x) \mid \int_X |u(x)|^p dx < +\infty \right\}, \quad 1 < p < \infty, \quad (6.2.3)$$

$$L^\infty(X) = \{u(x) \mid u: X \rightarrow \mathbb{R} \text{ 有界} \} \quad (6.2.4)$$

可以验证

$$d_p(u, v) = \left(\int_X |u(x) - v(x)|^p dx \right)^{1/p} \quad (6.2.5)$$

$$d_\infty(u, v) = \sup \{ |u(x) - v(x)| : x \in X \} \quad (6.2.6)$$

分别是 $L^p(X)$ 和 $L^\infty(X)$ 的度量.

定义 6.2.3 设 $\{x_n\}$ 是度量空间 (X, d) 的一个点列, 如果

$$\forall \varepsilon > 0, \exists N > 0, n, m > N \Rightarrow d(x_n, x_m) < \varepsilon, \quad (6.2.7)$$

则称点列 $\{x_n\}$ 是 Cauchy 点列 (或基本点列).

容易证明这样的命题: 收敛点列 $\{x_n\}$ 一定是 Cauchy 列. 有一类重要的度量空间, 在这样的空间中上述命题的逆也是成立的, 这就是所谓的“完备度量空间”. 在许多涉及极限的应用中, 往往要求度量空间是完备的. 例如, 后面将要介绍的压缩映象原理就必须在完备度量空间中讨论.

定义 6.2.4 如果度量空间 (X, d) 中的任意一个基本点列都收敛于 X 中的点, 就称 (X, d) 为完备度量空间, 并称 d 是完备度量.

可以验证, 例 6.2.1 和例 6.2.2 给出的度量空间都是完备度量空间. 此外, 在实数域 (或复数域), 基本点列必然收敛于实数 (或复数). 应该注意, 对于一般的度量空间却不能有这样的结果. 下面的例子表明, 确实存在这样的度量空间, 其中有不收敛的基本点列. 其实, 类似的例子还有很多.

例 6.2.3 设 X 是有理数集 \mathbb{Q} , 度量由两个有理数差的绝对值

$$d(r_1, r_2) = |r_1 - r_2|, \quad r_1, r_2 \in X \quad (6.2.8)$$

定义. 我们熟知度量空间 (\mathbb{Q}, d) 中存在着许多不收敛于有理数的基本点列. 例如 $\{(1 + \frac{1}{n})^n\}$ 就是一个, 已经证明它的极限 e (自然对数的底) 不是有理数. 因此, 这个有理数基本列不收敛于任何的有理数.

二、内积与范数

上一节介绍了度量的概念, 它统一了许多不同的收敛概念, 使应用中常常出现的不确切的“接近”、“逼近”、“误差”等说法精确化. 但是在许多应用领域, 只有

度量空间的概念还不够具体也不够用, 因为应用中常常需要所涉及的空间的元素之间存在某种代数关系. 例如, 对于图像空间来说, 度量仅仅给出了描述图像差异的误差测度. 然而对于图像处理, 往往要涉及图像能量、图像数据的相关性以及进行某种线性操作 (如叠加) 等运算. 因此, 从数学上看, 需要把图像空间看成一种向量空间, 并在其中定义图像的内积或范数. 因此, 讨论内积空间与赋范空间 (以及它们的完备化空间——Hilbert 空间与 Banach 空间) 是实际的需要. 为此, 必须首先定义向量空间 (线性空间) 的概念.

(一) 向量空间

向量与向量空间是泛函分析中最基本的概念之一, 也是学习分形图像编码的重要基础. 简单地说, 向量空间就是在非空集合中引进具有两种运算的一种线性结构, 即引进元素间的加法运算以及数与集合元素的乘法运算, 并规定它们必须满足一定的运算规则 (共八条, 请参考线性代数). 例如, \mathbb{R}^n 关于向量通常的加法和数乘运算就是一个向量空间. 在分形图像编码中, 这是最重要的一类向量空间.

(二) 内积

在通常的几何空间中, 向量有大小, 两向量间有夹角, 特别是两向量的正交性 (由此就有了勾股定理, 向量的投影等) 等重要概念. 这些重要概念在抽象向量空间中却没有得到反映. 在通常的 Euclid 几何空间中, 我们知道向量的大小、向量间的夹角都与向量的内积有密切的关系. 下面仿照通常的几何空间, 在向量空间中引入内积, 进而形成泛函分析中另一类重要的空间——Hilbert 空间.

在图像处理中, 内积可以刻画图像数据的相关性.

定义 6.2.5 设 H 是实向量空间, 如果任意 $x \in H, y \in H$, 都确定唯一的实数 $\langle x, y \rangle$, 满足条件:

- (1) 正定性: $\forall x \in H, \langle x, x \rangle \geq 0$, 而且 $\langle x, x \rangle = 0 \Leftrightarrow x = 0$;
- (2) 对称性: $\langle x, y \rangle = \langle y, x \rangle, \forall x, y \in H$;
- (3) 对第一变元的线性:

$$\langle kx + ly, z \rangle = k\langle x, z \rangle + l\langle y, z \rangle, \quad \forall x, y, z \in H, \quad \forall k, l \in \mathbb{R},$$

那么 $\langle x, y \rangle$ 称为 H 中的内积, 并把定义了内积的 H 称为内积空间.

内积空间 H 的度量由内积诱导:

$$d(x, y) = \sqrt{\langle x - y, x - y \rangle} \quad (6.2.9)$$

完备的内积空间称为 Hilbert 空间. 例如, 不难验证

$$\langle x, y \rangle = \sum_{i=1}^n x_i y_i \quad (6.2.10)$$

是 \mathbb{R}^n 的一种内积, 且 \mathbb{R}^n 关于这个内积是完备的. 这是最常用的内积 (Euclid 内积). 此外, 从内积可以诱导出向量长度、夹角、正交、投影等几何概念.

(三) 范数

从应用的观点看, 赋范空间是泛函分析中最重要的一种空间, 它是赋予了范数的向量空间. 范数是从通常向量长度的概念抽象出来的.

在图像处理中, 范数可以刻画图像的能量等. 下面是范数的定义.

定义 6.2.6 设 B 是实向量空间, 如果 B 上的实值函数 $p(x)$ 满足下列条件:

(1) 正定性: $p(x) \geq 0$, 而且 $p(x) = 0 \Leftrightarrow x = 0, \forall x \in B$;

(2) 齐次性: $p(kx) = kp(x), \forall x \in B, \forall k \in \mathbb{R}$;

(3) 三角不等式: $p(x+y) \leq p(x) + p(y), \forall x, y \in B$;

则称 $p(x)$ 是向量 x 的范数, 通常记为 $\|x\|$, 而且赋予了范数的 B 称为赋范空间. 完备赋范空间称为 Banach 空间.

显然, 内积空间是赋范空间, 范数由内积诱导出来:

$$\|x\| = \sqrt{\langle x, x \rangle}, \forall x \in H \quad (6.2.11)$$

而赋范空间 B 的度量由范数诱导:

$$d(x, y) = \|x - y\|, \forall x, y \in B \quad (6.2.12)$$

三、连续性与紧性

(一) 连续性

欧氏空间中与拓扑有关的概念 (如开集、闭集等), 能够以自然的方式推广到度量空间中去. 并进而仿照函数的连续性, 在度量空间中引入映射连续性的概念.

给定度量空间 (X, d) , 对于任意 $r > 0, x \in X$, 我们称

$$B(x, r) = \{y | d(x, y) < r, y \in X\} \quad (6.2.13)$$

为以 x 为心、 r 为半径的开球.

定义 6.2.7 (X, d) 为度量空间, A 是 X 的点集.

(1) 设 $x \in A$, 如果存在一个开球 $B(x, r) \subset A$, 那么称点 x 是点集 A 的内点.

(2) 若点集 A 的所有点都是内点, 则称 A 为开集. 空集规定为开集.

(3) 若点集 A 的补集 $X - A$ 为开集, 则称 A 是闭集.

定理 6.2.1 度量空间中的点集 A 为闭集的充分必要条件是 A 中任意收敛点列必收敛于 A 中的点. 即闭集关于极限运算是封闭的, 这也许是“闭集”的由来.

有了开集的概念, 现在我们就可以定义度量空间中点的邻域概念了.

定义 6.2.8 如果 X 是度量空间, x 是 X 一个任意的点, 则任意一个包含 x 的开集称为 x 的邻域.

下面引入映射连续性的这一重要概念, 它是通常函数连续性的自然推广.

定义 6.2.9 设 X 和 Y 是两个度量空间, $f: X \rightarrow Y$ 是任意映射. 称 f 在 $x \in X$ 是连续的, 如果对 $f(x)$ 的任何邻域 V , 存在 x 的邻域 U , 使得对所有的 $z \in U$, 都有 $f(z) \in V$.

若映射 $f(x)$ 在 $A \subset X$ 中的每个点连续, 则称它在 A 上是连续的. 当 Y 是实数直线或复数平面时, 称 $f(x)$ 为连续函数.

定理 6.2.2 设 X 和 Y 是两个度量空间, 映射 $f: X \rightarrow Y$ 在 $x \in X$ 连续的充分必要条件是对任何收敛于 x 的序列 $\{x_n\} \subset X$, 序列 $\{f(x_n)\}$ 收敛于 $f(x)$.

最后我们定义点到点集的距离. 在分形图像编码理论中, 这是一个重要的概念. IFS 逆问题就涉及这个距离概念.

定义 6.2.10 设度量空间 (X, d) , $x \in X$, $A \subset X$, 我们把 x 到 A 的距离定义为

$$\text{dist}(x, A) = \inf \{ d(x, y) \mid y \in A \}. \quad (6.2.14)$$

例如, 设 $x(t)$ 是闭区间 $[a, b]$ 上的连续函数, A 是阶数不超过 n 的多项式 $P_n(t)$ 的集合, 那么

$$\text{dist}(x, A) = \inf_{P_n \in A} d(x, P_n) = \inf_{P_n \in A} \max_{a \leq t \leq b} |x(t) - P_n(t)| \quad (6.2.15)$$

就是用阶数不超过 n 的多项式一致逼近 $x(t)$ 的最佳值.

(二) 紧性

我们在微积分中知道, 在实数直线情形, Weierstrass 定理 (或致密性定理) 与有限覆盖定理是两个重要的实数基本定理. 前者说任何有界数列存在收敛的子序列; 后者说, 覆盖有限闭区间的开覆盖 (有限开区间组成) 一定存在有限子覆盖 (有限个开区间组成). 这两个性质导出度量空间中 “紧集” 的概念. “紧性” 是拓扑空间的一个重要概念, 是化 “无限” 为 “有限” 的极为有用的性质.

定义 6.2.11 设 (X, d) 是度量空间, A 是 X 的子集称为紧的, 如果 A 包含在任何开集族 $\{A_i\}_{i \in I}$ 的并集之中, 则族中存在有限个开集 A_1, A_2, \dots, A_m , 使得 $A \subset \bigcup_{i=1}^m A_i$.

简言之, 称 A 是紧集, 如果 A 的每个开覆盖具有有限子覆盖. 紧性还有另外的等价刻画, 这里不加证明地给出其中的一个.

定理 6.2.3 完备度量空间 (X, d) 中的子集 A 是紧集, 当且仅当下列条件成立:

- (1) A 是闭的;

(2) A 中每个点列都有收敛子点列收敛于 A 中的一点.

由这个定理立即得出下面结论, 它在后面要用到.

定理 6.2.4 设 X 和 Y 是两个度量空间, $f: X \rightarrow Y$ 是连续映射, 则对 X 中的任意紧集 A , $f(A) = \{f(x) | x \in A\}$ 是 Y 中的紧集. 简言之, 连续映射保持紧性不变.

四、不动点定理

在数学中, 把一些方程的求解问题化为求映射的不动点以及用迭代法求不动点, 这是计算数学等数学分支中一个很重要的方法. 这个方法起源很早, 一直可以追溯到牛顿求代数方程根时所用的切线法, 后来 Picard 用迭代法求解常微分方程. 迭代法在不同领域中都有重要应用. 1922 年, Banach 把这个方法的基本点提炼出来, 用度量空间以及其中的压缩映射的一些基本概念更一般地描述了这个方法. 这种利用泛函分析研究方程解的近似方法以及关于映射不动点存在性的研究, 自 Banach 以后又取得了不少重要进展, 甚至成为非线性泛函分析的主要内容. 目前, 不动点原理的应用远不止求方程的近似解, 还有许多意想不到的应用. 分形图像编码就是不动点原理应用最成功的例子之一.

定义 6.2.12 度量空间 (X, d) 上的映射 T 称为压缩的, 如果存在 $\alpha \in [0, 1)$, 使得

$$d(Tx, Ty) \leq \alpha d(x, y), \quad \forall x, y \in X, \quad (6.2.16)$$

其中, α 称为映射 T 的压缩因子 (contractivity factor).

一个压缩映射总是缩短任意两点的映像之间的距离 (图 6-1). 映射 T 的不动点 x 是指满足 $Tx = x$ 的点, 即在 T 作用下 “不动” 的点.

定理 6.2.5 (压缩映射原理或 Banach 不动点定理) 设 (X, d) 是一个完备度量空间, $T: X \rightarrow X$ 是压缩因子为 $\alpha \in [0, 1)$ 的压缩映射, 则 T 存在唯一不动点 x_∞ , 且

$$x_\infty = \lim_{n \rightarrow \infty} T^n x = \lim_{n \rightarrow \infty} T(T^{n-1}x), \quad \forall x \in X. \quad (6.2.17)$$

换言之, 完备度量空间中压缩映射 T 的不动点存在唯一, 且可以由映射对任意初始点的反复迭代逼近到任意精度.

压缩映射原理是泛函分析的一个著名结果, 在泛函分析教科书或专著中都可以找到它的证明 (参看文献 [3,4]), 这里略去证明细节. 为了理解这个重要定理, 图 6-2 给出它的直观解释 (假设压缩因子为 0.5). 不妨视变换 T 为一辆 “汽车”, 它从任意初始地点 x 出发先到达 Tx , 行程为 $d(x, Tx)$, 接着从地点 Tx 到地点 T^2x , 行程为 $d(Tx, T^2x)$, 如此无限进行下去. 因为 T 的压缩因子为 0.5, 行程 $d(Tx, T^2x)$ 仅为行程 $d(x, Tx)$ 的一半. 一般地, 后一行程 $d(T^{n+1}x, T^{n+2}x)$ 始终只有前一行程

$d(T^n x, T^{n+1} x)$ 的一半. 可以想象, 如此无限进行下去, “汽车” 必将静止于某个地点 (“不动点”).

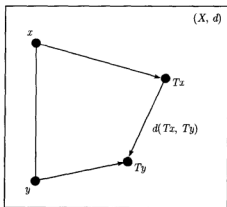


图 6-1 压缩变换的作用

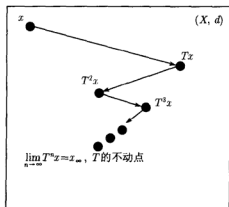


图 6-2 不动点定理的直观解释

应该指出, “压缩” 只是映射有不动点的充分条件而非必要条件. 论著 [4] 对此有详细讨论. 此外, 下面的例子表明, 定理 6.2.5 中压缩因子 α 不能放宽到 1 (例 6.2.1), 度量空间的完备性也不可少 (例 6.2.2).

例 6.2.4 设度量空间 $X = \mathbb{R}^+$, 其度量 $d(x, y) = |x - y|$ (绝对值), 映射 T 定义如下:

$$Tx = (1 + x^2)^{-1/2}, \quad x \in X \quad \text{或} \quad Tx = x + (x + 1)^{-1}, \quad x \in X \quad (6.2.18)$$

例 6.2.5 设度量空间 $X = (0, 1]$, 其度量 $d(x, y) = |x - y|$, 映射 T 定义为 $Tx = x/2$.

我们将知道, 分形图像解码基于压缩映射原理, 为了保证解码迭代收敛, 表达图像的分形变换一般被要求是压缩的. 但是, 实验结果表明^[9], 压缩因子可以增大到 1.5, 可见在分形图像编码中, 压缩性也仅仅是充分条件. 分形编码文献常常用 “最终压缩” (eventual contractivity)^[9] 的概念来解释这个现象. 映射 T 是最终压缩的, 是指对某个正整数 m , 映射 T^m 是压缩的. 这个术语常出现在分形编码文献中^[8].

最终压缩映射有下面的熟知性质 (下面的证明形式是作者给出的).

推论 6.2.1 设 (X, d) 是一个完备度量空间, 映射 $T: X \rightarrow X$. 如果存在一个正整数 m , 使 T^m 是压缩的, 则 T 存在唯一不动点 x_∞ , 并成立式 (6.17).

证 因为 T^m 是压缩的, 由定理 6.5, T^m 有唯一不动点, 记为 x_∞ . 于是 $T(x_\infty) = T(T^m x_\infty) = T^m(Tx_\infty)$. 这表明 $T(x_\infty)$ 也是 T^m 的不动点, 根据唯一性, $T(x_\infty) = x_\infty$. 此外, 设 $n > m, n = qm + r, 0 \leq r < m$, 根据定理 6.2.5 (以 $T^r x$ 为初始点), 得

到

$$x_{\infty} = \lim_{q \rightarrow \infty} (T^m)^q (T^r x) = \lim_{n \rightarrow \infty} T^n x, \quad \forall x \in X. \quad \text{证毕}$$

在分形图像编码技术中,一幅图像用一个压缩变换来编码,然后用源于定理 6.2.5 的迭代过程来解码.显然,定理 6.2.5 给出的迭代过程是不能控制的,于是现有分形解码的迭代解码过程也是无法控制的.为了实现对迭代解码过程的控制,作者在文献 [10] 中提出一个带控制参数的不动点定理 (定理 6.2.6),它给出了渐进逼近不动点的一个新颖迭代过程,迭代序列的收敛速度极大地受到这个控制参数的控制.特别地,当控制参数为 1 时,这个渐进不动点定理返回定理 6.2.5.

定理 6.2.6 (渐进 / 控制不动点定理)^[10] 设 X 是 Banach 空间, $T: X \rightarrow X$ 是压缩因子为 α 的压缩变换.那么变换 T 的唯一不动点 $\bar{x} \in X$ 可以通过下面的插值迭代得到:

$$x_{n+1} = (1 - \lambda)x_n + \lambda T x_n, \quad x_0 = x, \quad n = 0, 1, 2, \dots \quad (6.2.19)$$

也就是说, $\bar{x} = \lim_{n \rightarrow \infty} x_n$, 且不依赖于 x 和 λ . 其中, 控制参数 $\lambda \in (0, 1]$, x 是空间 X 中的任意向量.

证 设变换 $S: X \rightarrow X$ 定义为

$$Sx = (1 - \lambda)x + \lambda T x, \quad \forall x \in X,$$

则对于任意 $x, y \in X$, 我们有

$$\begin{aligned} \|Sx - Sy\| &= \|(1 - \lambda)(x - y) + \lambda(Tx - Ty)\| \\ &\leq \|(1 - \lambda)(x - y)\| + \|\lambda(Tx - Ty)\| \\ &\leq (1 - \lambda)\|x - y\| + \lambda\alpha\|x - y\| \\ &= s\|x - y\|, \end{aligned}$$

式中 $s = (1 - \lambda) + \lambda\alpha$. 因为 $\alpha, \lambda \in [0, 1]$, 所以 $s \in [0, 1]$, 于是 S 是压缩因子为 s 的压缩变换. 根据定理 6.2.5, 变换 S 有唯一不动点 $\bar{y} = S\bar{y}$. 注意到

$$\begin{aligned} S\bar{x} &= (1 - \lambda)\bar{x} + \lambda T\bar{x} \\ &= (1 - \lambda)\bar{x} + \lambda\bar{x} \\ &= \bar{x}, \end{aligned}$$

这表明 \bar{x} 也是变换 S 的不动点, 由唯一性即知 $\bar{y} = \bar{x}$. 证毕

基于定理 6.2.6, 作者在文献 [10] 中首次引入“渐进 / 可控分形解码”的概念, 并提出了一个渐进 / 可控分形解码算法, 它最大的优点是无需修改现有的分形编码过程, 因而无需特殊的分形编码器. 实验显示, 这个算法实现了对分形解码过程

的部分控制,可以控制分形解码速度的快慢,一次迭代的结果可以分多次迭代完成,从而实现了渐进分形解码.在某些应用场合,例如设计分形编码教学软件、利用计算机辅助技术制作卡通以及窄带渐进传输等方面,这个算法有较好的应用前景.我们将在第八章详细介绍这个算法.

五、拼贴定理

Banach 不动点定理表明,完备度量空间中的映射,只要是压缩的,它的不动点就存在且唯一,并可以由映射对任意初始点的反复迭代逼近到任意精度.然而,在分形图像编码中,必须解决其逆问题:给定一个点,寻求一个(非平凡^①)压缩变换,使它的不动点就是给定点.这是一个非常困难但极具挑战性的问题,就作者所知,至今尚未得到理想的解决.本节介绍的拼贴定理是对这个问题的部分回答.考虑到在泛函分析书中找不到拼贴定理,我们给出其详细证明.

引理 6.2.1 设 (X, d) 是度量空间. 对于固定的 $a \in X$, 函数 $F_a(x) = d(x, a)$ 是连续的.

证 根据三角不等式及对称性, 易知

$$d(x, z) - d(y, z) \leq [d(x, y) + d(y, z)] - d(y, z) = d(x, y),$$

$$d(y, z) - d(x, z) \leq d(y, x) = d(x, y), \quad \forall x, y, z \in X.$$

由此知

$$|d(y, z) - d(x, z)| \leq d(x, y), \quad \forall x, y, z \in X,$$

于是

$$|F_a(x) - F_a(y)| = |d(x, a) - d(y, a)| \leq d(x, y).$$

这就表明 $F_a(x)$ 关于 x 是连续的.

证毕

定理 6.2.7 (拼贴定理) 设 (X, d) 是一个完备度量空间, $T: X \rightarrow X$ 是压缩因子为 $\alpha \in [0, 1)$ 的压缩映射, 则 T 的不动点 x_∞ 满足:

$$d(x, x_\infty) \leq \frac{1}{1-\alpha} d(x, Tx), \quad \forall x \in X. \quad (6.2.20)$$

证 根据度量的连续性(引理), 我们有

$$\begin{aligned} d(x, x_\infty) &= d(x, \lim_{n \rightarrow \infty} T^n x) \\ &= \lim_{n \rightarrow \infty} d(x, T^n x) \\ &\leq \lim_{n \rightarrow \infty} \{d(x, Tx) + d(Tx, T^2 x) + \cdots + d(T^{n-1} x, T^n x)\} \end{aligned}$$

^① 应该指出, 给定待编码图像 x_0 , 对于任意图像 x , 定义变换 $T(x) = x_0$. T 显然是压缩的, 且 x_0 是它的唯一不动点. 但是, 这个变换仅仅具有理论意义, 因此, 我们把它称为平凡变换.

$$\begin{aligned}
&\leq \lim_{n \rightarrow \infty} \{d(x, Tx) + \alpha d(x, Tx) + \cdots + \alpha^{n-1} d(x, Tx)\} \\
&= \lim_{n \rightarrow \infty} \frac{1 - \alpha^n}{1 - \alpha} d(x, Tx) \quad (\text{因为 } 0 \leq \alpha < 1) \\
&= d(x, Tx) / (1 - \alpha).
\end{aligned}$$

证毕

定理 6.2.7 表明, 在完备度量空间 (X, d) 中, 要找一个压缩变换使其不动点与某个给定点 x_{org} 相近或相似, 即 $d(x_{\text{org}}, x_{\infty})$ 很小 (这是一个十分困难的极小化问题), 只需在度量空间 X 中找到一个压缩变换, 该变换使给定点近似不变, 即 $d(x_{\text{org}}, Tx_{\text{org}})$ 很小. 于是定理 6.2.7 把上述问题转换为“寻找给定点与其映像接近的压缩映射”的问题. 因此, 编码时只需极小化所谓的拼贴误差 $d(x_{\text{org}}, Tx_{\text{org}})$, 从而克服了极小化前一个距离的困难.

目前, 人们从不同的角度对拼贴定理进行了改进 (例如文献^[11~13]). 基于这些改进拼贴定理的分形编码器, 其编码性能都有不同程度的提高.

六、不动点的稳定性

在分形图像编码中, 压缩变换 (或其参数) 构成待编码图像的分形码. 寻找变换的过程不可能不产生误差. 此外, 在编解码系统中, 压缩变换的参数必须量化后存储或传输, 在解码端收到的分形码描述的变换只是一个“近似”的变换. 因此, 必须保证不会出现“差之毫厘, 失之千里”的情况. 为此, 我们提出压缩映射不动点的稳定性问题. 应该指出, 尽管在不动点理论的专著中找不到压缩映射不动点稳定性问题的讨论, 但在动力系统研究中, 吸引子的稳定性是一个重要的问题.

粗略地说, 如果变换 T 的微小变化不会引起不动点大的变化, 我们就说不动点是稳定的. 也就是说, 若压缩变换 T_1 和 T_2 “接近”, 则它们的不动点也“接近”. 反映在分形图像编码中, 分形码 (变换参数) 的微小变化不会引起重构图像大的误差, 我们就说分形码是稳定的.

设 (X, d_X) 是紧度量空间, $\text{Con}(X)$ 表示 X 上的压缩变换的集合. 定义:

$$d_C(T_1, T_2) = \sup_{x \in X} d_X(T_1 x, T_2 x), \quad T_1, T_2 \in \text{Con}(X). \quad (6.2.21)$$

容易验证, 它同时满足三条度量公理, 因此, $(\text{Con}(X), d_C)$ 是度量空间.

下面的定理表明, 若压缩变换 T_1 和 T_2 “接近”, 则它们的不动点也“接近”.

定理 6.2.8 (稳定性)^[14] 设 (X, d_X) 是紧度量空间, T_1, T_2 是压缩因子分别是 s_1, s_2 的压缩映射, 则

$$d_X(\bar{x}_1, \bar{x}_2) \leq \frac{1}{1-s} d_C(T_1, T_2), \quad (6.2.22)$$

其中 $s = \min(s_1, s_2)$, \bar{x}_i 是变换 $T_i (i=1, 2)$ 的不动点.

证 证明很简单. 因为

$$\begin{aligned}
 d_X(\bar{x}_1, \bar{x}_2) &= d_X(T_1\bar{x}_1, T_2\bar{x}_2) \\
 &\leq d_X(T_1\bar{x}_1, T_1\bar{x}_2) + d_X(T_1\bar{x}_2, T_2\bar{x}_2) \\
 &\leq s_1 d_X(\bar{x}_1, \bar{x}_2) + d_C(T_1, T_2), \\
 d_X(\bar{x}_1, \bar{x}_2) &= d_X(T_1\bar{x}_1, T_2\bar{x}_2) \\
 &\leq d_X(T_1\bar{x}_1, T_2\bar{x}_1) + d_X(T_2\bar{x}_1, T_2\bar{x}_2) \\
 &\leq d_C(T_1, T_2) + s_2 d_X(\bar{x}_1, \bar{x}_2),
 \end{aligned}$$

所以,

$$d_X(\bar{x}_1, \bar{x}_2) \leq d_C(T_1, T_2) + \min(s_1, s_2) d_X(\bar{x}_1, \bar{x}_2).$$

由此推出式 (6.2.22).

证毕

第三节 分形

分形理论是非线性科学研究中一个十分活跃的分支, 特别是近十余年来在计算机图像处理和分析中已得到广泛应用. 同时, 要了解分形图像压缩技术的起源, 以及理解该技术的数学原理, 分形的概念是必不可少的.

分形源于数学家 Mandelbrot 对棉花价格变化规律的研究 (1960), 他发现一天中的棉花价格变化曲线与一月中的棉花价格变化曲线几乎是相同的, 这种尺度不变性启发他去探索大自然的尺度现象. 后来, Mandelbrot 去了 IBM 公司并研究通讯传输中的噪声问题, 结果发现尺度不变性现象也存在于这一领域, 无论是以秒为尺度观察还是以小时为尺度观察, 无噪声时段与噪声时段之比总是一个常数. 接着 Mandelbrot 对“英国海岸线有多长?”这个貌似简单、其实非常复杂的问题进行了深入思考与研究, 结果他发现英国海岸线的长度依赖于测量所用的尺度这一惊人的事实! 我们知道, 要得到自然界一条实际曲线 (如海岸线) 的长度, 若不知道这条曲线的函数形式, 唯一的方法就是对其进行测量. 按照传统几何的观点, 长度是不会依赖于测量所用尺度的. 但是, 海岸线经历了长期的冲刷、侵蚀、沉积等随机地质构造运动, 形状非常复杂, 局部放大后可以发现“海湾包含小海湾、半岛里面有半岛”, 无论用尺度多么小的尺子去测量海岸线, 更小尺度的“海湾”、“半岛”都将被忽略, 因此, 测量的结果总是不确定的, 且尺度越小, 长度越大.

1977 年, 在前述研究以及总结前人成果的基础上, Mandelbrot 提出了“分形”的概念, 出版了划时代专著^[15]. 他在此专著中第一次系统地阐述了分形的思想、内容、意义和方法. 这本专著的出版标志着分形几何作为一个独立的数学分支正式诞生. 后来, Mandelbrot 出版了另一本经典著作^[16]. 从此, 大批来自数学、物理、化学、生物、电子信息、材料科学、地质以及社会科学等学科的学者们, 纷纷进入

了分形的研究领域,使分形研究空前活跃.他们对分形概念做出了各种各样的研究和分析,特别是分形理论的研究,使一些原已死寂一般的老学科焕发了生机.尽管分形的数学理论还没有形成完善的理论体系,但是全新的分形思维方式改变了人们认识自然的观念.

本节仅仅简单介绍分形的基本思想以及在图像处理中的应用概况.

一、分形概念

说到分形,人们会列举一大堆描述它的基本特性,如任意尺度下都显示细节,有某种自相似结构,“维数”可以是分数,等等.但究竟什么是“分形”?恐怕谁也说不清楚,因为目前还没有公认的严格明确的定义,只能对它的特征进行一些描述^[17].

粗略地说,分形是对没有特征长度但具有一定意义下的自相似图形和结构的总称,这里特征长度可以理解为刻画一个几何体特征的长度(如直径就是一个球的特征长度).Mandelbrot构造的“fractal”(分形)一词就是取“破碎的,不规则的”之意,它描述了不同于欧氏几何所说的几何体.他曾建议将“分形”定义为整体与局部在某种意义下具有自相似性(self-similarity)的集合,也曾把“分形”定义为 Hausdorff 维数^[17]严格大于拓扑维数^①的集合.但是这些定义都不够精确、不够全面,因为按照这些定义,有些明显应算是“分形”的集合被排除在外.

英国分形几何学家 Falconer 认为^[17],“分形”的概念可以按生物学中对“生命”概念的类似方法处理.在生物学中“生命”并没有严格和明确的定义,但却可以列出一系列生物体的特征,如繁殖能力、运动能力以及对周围环境的相对独立的存在能力等等.虽然有一些生物例外,但大部分生物都具有上述特征.同样,我们也可以不寻求分形的确切简明的定义,而寻求分形的特性.于是,分形可以看作是具有或部分具有下列典型性质的复杂集合^[17]:

- (1) 具有精细的结构,即有任意小尺度比例的细节,或任意尺度下都显示细节;
- (2) 非常不规则,它的整体和局部都不能用欧氏几何来描述;
- (3) 通常有某种自相似的结构,可能是近似的或是统计的;
- (4) 一般地,以某种方式定义的“分形维数”大于它的拓扑维数;
- (5) 在大多数令人感兴趣的情形,可用非常简单的方法定义,也可能迭代产生.

以上性质中,自相似性与分形维数是分形的两大显著特征.为了对分形有个直观理解,我们先看几个典型分形.

例 6.3.1 Koch 曲线 (图 6-3)

^① 一种在拓扑变换之下保持不变的维数.拓扑变换可以直观理解为既不能变一点为两点、又不能变两点为一点的连续形变.

从平面上一单位线段 (图 (a)) 开始, 第一步, 将单位线段三等分, 将中间部分用两条长 $1/3$ 的线段来替代 (图 (b)); 第二步, 将图 (b) 中的每条线段三等分, 中间的一段用长 $1/3^2$ 的两条线段替代, 得到图 (c); 重复这个迭代过程得到图 (d); 不断重复这样的迭代作法, 无穷次迭代后就生成了具有处处连续、处处不可微的 Koch 曲线 (图 (e)).

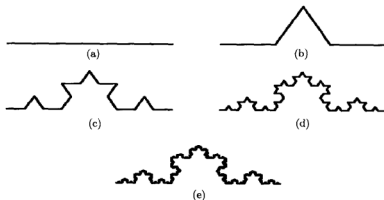


图 6-3 Koch 曲线构造的不同阶段

例 6.3.2 Cantor 三分集 (图 6-4)

从平面上一单位线段 ($n=0$) 开始, 第一步, 将单位线段三等分, 将中间部分去掉 ($n=1$); 第二步, 将余下的每条线段三等分, 都去掉中间的一段 ($n=2$); 重复这个迭代过程得到图 $n(n=3, 4, 5)$; 不断重复这样的迭代作法, 无穷次迭代后就生成了 Cantor 三分集 (无法给出图示).

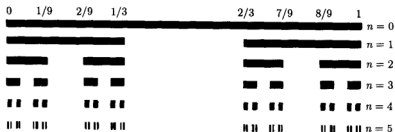


图 6-4 Cantor 三分集构造的不同阶段

例 6.3.3 Sierpinski 三角 (图 6-5)

在图 6-5 中, 上图是多功能缩印机示意图。这台缩印机可以看成是具有下列特性的复印机: 具有多个镜头, 可以产生原图像的多个可重叠的拷贝; 每个镜头缩小原图像的尺寸; 以反馈方式工作, 每次复印输出作为下一次复印的输入。初始输入

图像可以任意选择. 下图是起始于作者姓名的七次缩印的不同阶段, 不断重复这样的迭代缩印, 无穷次迭代后就生成了 Sierpinski 三角 (图 6-5 中下图).

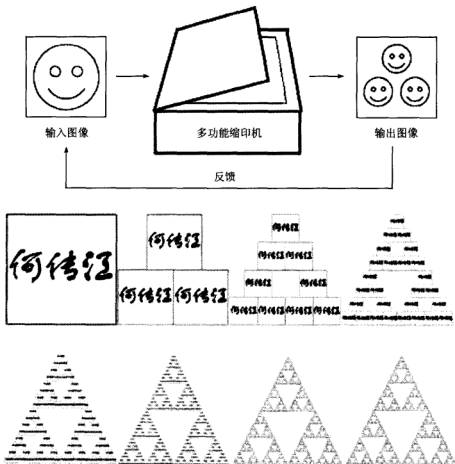


图 6-5 Sierpinski 三角构造的不同阶段 (上图是多功能缩印机示意图)

下面回到分形的两大显著特征 —— 自相似性与分形维数的讨论.

自相似性是分形的显著特征, 也是分形几何的主要概念之一. 所谓自相似性, 是指无论几何尺度如何变化, 分形的部分与部分之间或部分与整体之间具有某种意义下的相似性. 它是分形有别于欧氏几何形体的一个重要特征. 这种尺度不变性 (scale invariance) 在大自然中随处可见. 这是分形广泛应用于自然科学、社会科学诸多领域的客观基础. 我们将会看到, 分形图像压缩技术正是利用了自然景物图像中存在的自相似性而获得了高压缩比. 可以毫不夸张地说, 自相似性也是分形图像压缩技术的精髓.

为了研究 Koch 曲线等分形, 我们必须接受一个重要的概念, 这就是特征尺度

(characteristic size). 什么是特征尺度呢? 我们仅仅作个直观的理解. 例如高空拍照时, 一定得有一个能表示尺度大小的物体作为一种特征尺度, 没有这些物体, 很难确定这些照片反映的是 10 米方圆还是 10 公里范围的景象. 假如我们有一幅海岸线照片 (Koch 曲线是海岸线的理想数学模型), 但照片中并没有表示特征长度的物体 (如一棵树), 那么就很难确定照片是从多高的地方拍摄的. 这是因为海岸线 (Koch 曲线) 具有自相似性, 即将海岸线 (Koch 曲线) 的一部分放大后, 它与原来的海岸线看起来是相似的.

传统几何的研究对象一般总有自己的特征尺度, 可用恰当的尺度去测量. 但用一把固定的尺子去测量 Koch 曲线就会遇到困难, Koch 曲线是不具有特征尺度的几何对象, 无论用多短的尺子去测量它, 都难以得到它的确切长度. Koch 曲线具有无穷多折线, 当测量尺子的尺度越来越小时, 新的结构却与整体结构具有同样的复杂性, 因为放大 Koch 曲线的任意部分, 其形状与整体相同. 因此, 测量的结果总是不确定的, 且尺度越小, 长度越大. 可见, 没有特征尺度的核心是自相似性, 它对应于分形中存在着的层次结构 (hierarchy).

维数是几何体的重要特征, 是一种最基本的几何不变量. 在欧氏几何中, 维数定义为描述空间中一个点的位置所需要的独立坐标数目或连续参数的最小数目, 于是, 曲面是 2 维的, 曲线是 1 维的, 点是 0 维的. 这种维数在一定程度上描述了一个点集占有空间的规模. 然而, 按照这个维数定义去计算填充平面的 Peano 曲线的维数, 不难得得出两个互相矛盾的答案——作为曲线, 其维数是 1, 但它填满平面, 故维数是 2. 尽管前苏联数学家乌雷松定义的维数^①避免了这种矛盾, 比欧氏维数前进了一步, 但是这种拓扑维数也没有对点集占有空间的规模 (或点集的整体复杂程度) 给出很好的描述. 其实, 在圆锥曲线和 Peano 曲线之间还存在许许多多的曲线 (如 Koch 曲线), 它们的复杂程度很不一样, 占有空间的规模也大不相同. 但只要没有填充一个邻域, 按乌雷松的定义, 它们的维数都为 1. 于是, Koch 曲线是一维的, 这显然没有刻画出 Koch 曲线的复杂性和占有空间的规模远大于圆锥曲线的事实. 因此, 为了刻画 Koch 曲线等复杂曲线占有空间的规模, 我们需要进一步扩充维数的概念, 放弃维数必须是整数的固有观念. 例如, 容许曲线维数取 1 与 2 之间的实数, 曲面维数取 2 与 3 之间的实数, 等等. 引进非整数维数, 这是分形几何刚出现时最令人困惑的一点, 但许多应用实例都表明这种整维数概念延拓的合理性. 目前, 分形维数是一种比较分形的客观工具, 是目前刻画分形不规则性的一个重要指标, 在图像处理等领域中有广泛应用 (详见后文).

分形维数目前已有多种定义, 大量能够用于估计分形维数的方法都已在 Mon-

^① 乌雷松是维数理论的创立者, 其维数定义如下: 如果不存在包含多于一个点的连通图形, 则说图形是 0 维的; 归纳地, 在已经确定 $n-1$ 维和更低维图形的前提下, n 维图形定义为不是 $n-1$ 维或更低维的, 且可以用 $n-1$ 维 (或更低维) 的图形把其中任意点及其邻近点同图形的其余部分分割开. 按此定义, 填充平面的 Peano 曲线是 2 维的.

delbrot 的专著^[16]中讨论。目前,分形维数已成为分形几何的主要工具和研究对象。在分形维数的多种定义中, Hausdorff 维数、自相似维数以及盒维数是三种重要的维数。

Hausdorff 维数是建立在 Hausdorff 测度基础上的一种分形维数^[17],是分形几何维数理论的基础,也是理论较为完善的一种维数。但是,计算一个分形的 Hausdorff 维数往往是复杂而又困难的,这种计算上的困难极大地限制了 Hausdorff 维数的实际应用。

像 Cantor 集、Koch 曲线和 Sierpinski 三角这样的分形,其任一局部的形状与整体是相似的,只要将每一个局部放大一定倍数就可得到与整体一致的图形。这种集称作严格自相似集。自相似维数给出了对这种自相似集的共同数学描述。

在许多情况下,人们代之以其他简便实用的维数。盒维数(box-counting dimension)就是其中一个具有广泛应用的维数^[18],它有简便、能通过实验手段近似地计算且对于有无自相似的图形(pattern)都实用的优点^[19]。

盒维数定义如下:取尺寸为 d 的小盒子(如拓扑维为 D 的小超立方体、小球等),把分形覆盖起来。由于分形内部有各种层次的空洞和缝隙,所以有些小盒子是空的,有些小盒子覆盖了分形的一部分。记非空小盒子的最小数目为 $N(d)$ 。然后缩小盒子的尺寸 d , 所得 $N(d)$ 自然要增大。当 $d \rightarrow 0$ 时,得到数盒子法定义的分形维数

$$D = -\lim_{d \rightarrow 0} \log N(d) / \log d. \quad (6.3.1)$$

实际应用中, d 只能取有限值,因此,通常的作法是求一系列的 d 和 $N(d)$, 然后由双对数坐标系 $\log N - \log d$ 中回归直线的斜率求 D 。回归直线的方程为

$$\log(N(d)) = \log(K) + D \log(1/d), \quad (6.3.2)$$

其中 K 是正常数。显然, $N(d)$ 与 $(1/d)^D$ 成比例(幂律^[16])。易求得 Koch 曲线的盒维数

$$D = \frac{\ln 4^n}{\ln 3^n} = \frac{\ln 4}{\ln 3} \approx 1.26 > 1 = D_T \quad (D_T \text{ 是拓扑维数}),$$

它正好等于 Koch 曲线的 Hausdorff 维数^[17]。

分形维数 D 的值指明了分形的复杂程度,或给出了集合粗糙程度的一个度量,集合越粗糙, D 的值越大^[18,20]。它的主要缺点是没有反映几何对象的不均匀性,含有一个点和许多点的盒子在式 (6.3.1) 中具有同样的权重(关于它的改进,详见文献^[18])。此外,在实际应用中,一个自然集合(natural set)被认为是分形,如果 D 在很宽的尺度范围内是稳定的^[21]。

总之,分形是人们在自然界和社会实践活动中所遇到的不规则事物的一种数学抽象。人们对于分形的兴趣源于它可以描述和解决一些实际问题,正如历史上人

们对于欧氏几何与微积分的应用一样,这种描述和应用允许在一定尺度下的近似性。同样,在利用分形来描述海岸线、云层的边界、地表的形状、岩石的裂缝、流体的湍流以及一些经济现象时,也具有在一定意义下的近似性。实际上,现实世界中没有真正的分形,犹如现实世界中没有真正的圆一样。正如 Mandelbrot 所强调的那样,自然界的分形与数学中讨论的分形是有区别的。

二、分形在图像处理中的应用

从诞生的那天起,分形理论就与计算机的发展息息相关,相辅相成。一方面,分形理论推动了计算机可视化技术的迅猛发展,并使计算机在信息压缩、存储及模拟自然现象中的各种奇妙图形发挥了重要作用;另一方面,分形研究极大地依赖于计算机技术的发展,计算机技术为这门学科的研究提供了强有力的工具。计算机的直观表述能力,促使这门学科从纯粹的基础科学拓展到应用科学的范畴。此外,利用分形理论模拟自然景观展现出的一大批优美图案,促使分形理论与计算机科学理论的融合与发展。

自然界的分形随处可见,绝非个别现象,这是分形在许多应用领域取得巨大成功的客观基础。自 Mandelbrot 创立分形概念以来,分形理论已广泛应用于数学、物理学、化学、地球物理学、植物学、生物学、计算机图形学、计算机视觉以及图像处理等众多学科中。分形的思想和方法也日益影响着现代社会的生与活动。虽然分形不是理解所有现象的灵丹妙药,但在许多不同类型的分析中,分形的确能给我们帮助。

在计算机图形学领域,具有统计自相似性的分形是一个非常有诱惑力的对象。借助于这种统计分形的概念,一方面,用十分简单的规则建立非常复杂的模型,可生成了大量酷似自然景观的图形;另一方面,给定一个复杂的现象,可以将它作为简单规则在不同尺度上的重复作用(迭代)所产生的结果,对它进行分析甚至加以描述,从而十分逼真地模拟出来。

在计算机视觉领域,分形及其维数也扮演了重要的角色。已有研究表明,许多确定性或统计分形与自然景物看起来十分相似(如 Barnsley' fern 与天然 fern)。这是分形广泛应用于计算机图形学的视觉基础。此外,实验表明,集合的分形维数与人们对它的粗糙感有密切的联系^[20]。虽然据此要生成一个集合在视觉上好的近似仅分形维数是不够的,但是分形维数确实可以作为生成规则的一个参数^[21,22]。另外,依据人类视觉系统对图像区域任意部分细节的敏感性依赖于周围背景的活动程度(amount of activity),图像区域的分形维数已作为这种活动性的一个客观度量^[23]。

在图像处理领域,分形理论已成功地应用于图像压缩。图像分形压缩方法的创始人 Barnsley 的迭代函数系统是最引人注目的方法,论著《处处分形》^[1]总结了在 1985~1993 年期间他(或与合作者)近 20 篇论文的内容及其他学者的研究成

果。他的迭代系统公司(美国)已经开发出了采用分形图像压缩技术的硬件产品,被认为是目前数学理论与数学方法直接创造经济效益的最有影响的事件之一。

除迭代函数系统外,还有其他几种基于分形的方法,它们基本上是彼此不同的。下面是它们极简短的介绍(本书不再讨论它们)。

分形曲线,特别是 Peano 曲线(一种填满平面的曲线)已被用于图像扫描,取得了比标准的光栅扫描(raster scanning)方式更好的效果^[24]。分形维数已应用于自然景观的图像建模、图像分割和特征提取^[25~27],也用于区分同一背景下的不同自然物体,描述人造物体与自然物体的差异^[27]。其依据之一是自然物体的分形维数彼此不等、也不等于人造分形的维数。许多研究者把分形维数用于图像分割^[20,27~31],例如 Pentland^[20]把分形维数作为分割图像的一个参数;Jang 和 Rajala^[30,31]将图像分割以后,把分形维数作为图像块(segment)复杂程度的一种度量,并根据复杂程度决定图像块的编码方法。图像纹理是图像分析与处理的一个重要内容,文献^[28,29,32~35]把分形维数应用于图像纹理的研究中。纹理是物体表面不规则程度的一种度量,而图像的分形维数正好反映了这种变化,它与人类视觉系统对图像纹理粗糙度的感知是一致的。分形维数越大,对应的图像表面越粗糙;分形维数越小,对应的图像表面越光滑。因此,准确计算作为图像纹理特征的分形维数就可以较好地对图像进行分割,从而有利于场景分析。在智能机器人等研究领域,对场景分析的实时性要求是必要的,故图像分形维数的实时计算是一个十分重要的问题。Goel 和 Kwatra^[23]把分形维数用在分形图像编码器中以调整误差阈值。Cheong 等^[36]则利用分形维数选择多尺度边缘探测器(multiscale edge detector)的最优尺度参数。在这种方法中,边缘点由小波变换探测,而缩放(dilation)参数则由分形维数控制。图像轮廓的分形编码是 IFS 最早的应用之一^[37],后来,Jacobs 等人^[38]也把类似的方法用于图像轮廓的分形编码。在国际互连网上,可以得到许多分形编码的资料和文献,这说明分形图像压缩技术是目前国际上十分活跃的研究领域之一。

总之,随着分形理论的深入研究和计算机算法的进一步完善,分形方法必将在工程实践中发挥越来越大的作用。正如物理学家 J. A. Wheeler 所说:“明天谁不熟悉分形,谁就不能被认为是科学上的文化人。”需要指出,当代科学对分形的研究仍处于具体分析阶段,尚未奠定统一的理论基础。因而对它们的深入研究还有待于科学的进一步发展。对这些问题进行广泛、深入和细致的研究,无论在理论上还是在造福人类的应用上都具有重大而深远的意义。

第四节 迭代函数系统

迭代函数系统是分形图像压缩编码技术的理论基础,著名学者 Mandelbrot、Hutchinson、Barnsley 和 Jacquin 等为它的产生和应用做出了重要贡献。

Mandelbrot 首先揭示了分形的本质特征, 确定了分形几何的理论框架^[15,16]. 之后, 分形的研究超出数学的范畴, 受到众多学科领域研究者的极大关注. 目前, 分形理论 (核心是分形几何) 已成为探讨复杂现象与无序现象的强有力的数学工具, 被誉为 20 世纪后期非线性科学的三大理论突破之一.

Hutchinson 1981 年提出迭代函数理论^[2], 对自相似集进行了深入研究, 并证明了迭代函数的吸引子为分形. 尽管他没有给出任何的技术应用, 但为 Barnsley 等创立分形图像编码奠定了理论基础.

Barnsley 1988 年发表分形图像编码的奠基性著作^[1], 从此分形技术被引入计算机图形学. 他在书中提出了“迭代函数系统”和“拼贴定理”, 详细论述了它们的数学理论, 并成功地用来模拟自然景观 (如 Barnsley' fern). 当时, Barnsley 和他的研究小组发现用简单的迭代函数系统就可生成与自然景物相似的、具有无限细节的复杂图形, 由此看到迭代函数系统建模某些具有分形特征的自然景观 (如云彩、树叶、山脉、海岸线等) 的潜力, 并用迭代函数系统逼真地再现了这些自然分形图形. 随后他们又提出用迭代函数系统压缩编码图像的思想, 尽管提出的方法被证明是不实用的, 但毕竟迈出了分形图像压缩编码最艰难的第一步.

在分形压缩编码实用化的研究中, Jacquin 把 Barnsley 的迭代函数系统编码算法作了实质性修改, 在其 1992 年发表的获奖论文^[39]里首次提出了分形块编码 (fractal block coding), 它是一种基于方块划分的分形压缩编码的计算机自动算法. Jacquin 迈出了分形图像编码实用化的第一步, 使分形图像编码的研究重新活跃起来, 也为分形编码指明了一条发展之路. 事实上, 正是在分形块编码算法出现之后, 分形图像编码才真正成为一个活跃的研究领域. 此外, 现在的分形编码基本上都是基于分形块编码的方案^[8].

现在, 用各种方法在计算机上生成的分形结构越来越多^[40]. 但是, 到目前为止, 用一个数学的系统去解析地构造、研究一大类存在于人造的或自然的、具有“自记忆”(自仿射相似) 结构的分形, 最为成功的还是广为人知的迭代函数系统, 它既包含确定性的过程也包含随机的过程. 本节中, 我们对迭代函数系统作个初步介绍, 其中除要涉及一些集合论、拓扑学的基本概念外, 许多定理的证明都比较抽象、繁冗. 篇幅所限, 这里叙述的多数结论, 不能够给出证明细节. 好在许多分形几何书中可以找到结论的证明.

一、分形空间

和许多其他几何问题一样, 分形问题的研究也必须在某个合适的拓扑空间中进行. 从理论与应用两方面看, 完备度量空间是最佳的选择.

设 (X, d) 是一个度量空间, 拓扑由度量诱导. X 的非空紧子集的全体构成一个集合, 记为 $\mathcal{H}(X)$. 不熟悉拓扑学的读者, 不妨把紧子集理解为有界闭集, 因为

应用中 X 通常是欧氏空间 \mathbb{R}^n (在 \mathbb{R}^n 中, 紧集与有界闭集是等价的).

定义 6.4.1 设 $x \in X, B \in \mathcal{H}(X)$, 我们把 x 到 B 的距离定义为

$$\text{dist}(x, B) = \inf \{ d(x, y) \mid y \in B \}. \quad (6.4.1)$$

显然, $x \in B$ 时, $\text{dist}(x, B) = 0$. 作为 x 到 B 的距离, 自然要求 $\text{dist}(x, B) > 0$, $\forall x \notin B$. 为此, 我们证明下面的定理.

定理 6.4.1 存在 $x_0 \in B$, 使得 $\text{dist}(x, B) = d(x, x_0)$. 从而当 $x \notin B$ 时, $x \neq x_0$, $\text{dist}(x, B) = d(x, x_0) > 0$.

证 考察 y 的函数: $f: B \rightarrow \mathbb{R}, f(y) = d(x, y), y \in B$. 由度量的定义, 易知 $f(y)$ 是连续函数. 因为 B 是非空紧子集, 所以 $f(y)$ 在 B 上取得最小值 m , 且 $m = \inf \{ f(y) \mid y \in B \}$. 由下确界定义可知, 存在点列 $\{y_n\} \subset B$, 使得

$$|f(y_n) - m| < \frac{1}{n}, \quad n = 1, 2, \dots$$

再次用到 B 的紧性, 点列 $\{y_n\}$ 有子列 (仍记为本身) 收敛于 $x_0 \in B$. 根据 f 的连续性即知 $f(x_0) = m$, 即 $\text{dist}(x, B) = d(x, x_0)$. 证毕

对于任意 $A, B \in \mathcal{H}(X)$, 记^①

$$\rho(A, B) = \sup \{ \text{dist}(x, B) \mid x \in A \}.$$

容易举例说明, $\rho(A, B)$ 一般不满足正定性与对称性, 因此不能作为 $\mathcal{H}(X)$ 的度量. 为此, 自然考虑定义:

$$d_H(A, B) = \max\{\rho(A, B), \rho(B, A)\}, \quad A, B \in \mathcal{H}(X) \quad (6.4.2)$$

定理 6.4.2 d_H 满足度量的三条性质 (正定性、对称性与三角不等式), 从而 d_H 是 $\mathcal{H}(X)$ 的一个度量.

证 正定性、对称性是显然的, 我们仅给出三角不等式的验证, 即证明:

$$d_H(A, B) \leq d_H(A, C) + d_H(C, B), \quad \forall A, B, C \in \mathcal{H}(X). \quad (6.4.3)$$

为此, 我们先断言 $\rho(A, B) \leq \rho(A, C) + \rho(C, B)$.

^① 在文献中 $\rho(A, B)$ 称为紧子集 $A \in \mathcal{H}(X)$ 到紧子集 $B \in \mathcal{H}(X)$ 的距离. 这容易引起误会, 因为在数学中, “距离” (度量) 必须满足正定性、对称性与三角不等式这三条距离公理. $\rho(A, B)$ 显然不满足正定性与对称性.

事实上, 对于任意 $x \in A$,

$$\begin{aligned} \text{dist}(x, B) &= \inf \{ d(x, y) \mid y \in B \} \\ &\leq \inf \{ d(x, z) + d(z, y) \mid y \in B \} \\ &= d(x, z) + \inf \{ d(z, y) \mid y \in B \} \quad (\forall z \in C) \\ &\leq d(x, z) + \text{dist}(z, B) \\ &\leq d(x, z) + \rho(C, B). \end{aligned}$$

上式对 z 取下确界, 得到

$$\begin{aligned} \text{dist}(x, B) &\leq \inf \{ d(x, z) \mid z \in C \} + \rho(C, B) \\ &\leq \text{dist}(x, C) + \rho(C, B). \end{aligned}$$

上式对 x 取上确界即得

$$\rho(A, B) \leq \rho(A, C) + \rho(C, B).$$

同理证明: $\rho(B, A) \leq \rho(B, C) + \rho(C, A)$, 结合上述不等式即得三角不等式 (6.4.3).

证毕

以后我们称 $d_H(A, B)$ 为紧子集 $A, B \in \mathcal{H}(X)$ 的 Hausdorff 距离 (度量).

定理 6.4.3 当 (X, d) 是完备度量空间时, $(\mathcal{H}(X), d_H)$ 也是完备度量空间.

这个定理的证明较繁, 这里略去. Barnsley 认为 $(\mathcal{H}(X), d_H)$ 是分形所在的空间, 分形之间的距离也正是由这种 Hausdorff 距离度量的. 因此, 把这个度量空间称为分形空间^①.

二、分形空间上的压缩映射

分形空间 $\mathcal{H}(X)$ 是由 X 的非空紧子集构成一个集合, 如何从映射 $w: X \rightarrow X$ 出发构造 $\mathcal{H}(X)$ 上的映射? 按自然的方式, 应该定义

$$w(A) = \{w(x) \mid x \in A\}, \quad \forall A \in \mathcal{H}(X). \quad (6.4.4)$$

为了说明这确实定义了 $\mathcal{H}(X)$ 到 $\mathcal{H}(X)$ 的一个映射, 自然要问下面的问题是否成立:

$$\forall A \in \mathcal{H}(X) \Rightarrow w(A) \in \mathcal{H}(X).$$

为此, 我们先介绍拓扑学中的两个相关结论.

^① 有的文献把 $(\mathcal{H}(X), d_H)$ 称为 Hausdorff 空间, 这容易与拓扑学中的 Hausdorff 空间 (T_2 空间) 相混.

引理 6.4.1 设 (X, d) 是一个度量空间, 拓扑由度量诱导, 以及映射 $w: X \rightarrow X$.

(1) 如果 w 是压缩的, 则 w 是连续的;

(2) 如果 w 是连续的, 则映射 w 保持紧性不变, 即把 X 的紧子集变成紧子集.

证 结论 (1) 从 w 的压缩性直接导出; 结论 (2) 由定理 6.2.4 得到. 证毕

根据上述引理, 对于任意 $A \in \mathcal{H}(X)$, $w(A) \in \mathcal{H}(X)$. 因此, 压缩映射 $w: X \rightarrow X$ 按自然的方式诱导出 $\mathcal{H}(X)$ 上的映射:

$$\begin{aligned} w: \mathcal{H}(X) &\rightarrow \mathcal{H}(X), \\ A &\mapsto w(A), \end{aligned} \quad (6.4.5)$$

其中, $w(A) = \{w(x) \mid x \in A\}$, $\forall A \in \mathcal{H}(X)$.

这里, 我们用同一字母 w 表示 X 上的映射, 又表示 $\mathcal{H}(X)$ 上的映射, 这不会引起混乱, 因为从上下文易辨其意.

定理 6.4.4 设 $w: X \rightarrow X$ 是度量空间 (X, d) 上压缩比为 s 的压缩映射, 则 $w: \mathcal{H}(X) \rightarrow \mathcal{H}(X)$ 是分形空间 $(\mathcal{H}(X), d_H)$ 上压缩比为 s 的压缩映射.

证 设 $A, B \in \mathcal{H}(X)$, 任取 $y_1 \in w(A)$, 设 $x_1 \in A$ 使得 $y_1 = w(x_1)$,

$$\begin{aligned} \text{dist}(y_1, w(B)) &= \inf\{d(y_1, y_2) \mid y_2 \in w(B)\} \\ &= \inf\{d(w(x_1), w(x_2)) \mid x_2 \in B\} \\ &\leq \inf\{s \cdot d(x_1, x_2) \mid x_2 \in B\} \\ &= s \cdot \text{dist}(x_1, B) \\ &\leq s \cdot \rho(A, B). \end{aligned}$$

上式对 y_1 取上确界, 得到

$$\rho(w(A), w(B)) \leq s \cdot \rho(A, B).$$

同理可证:

$$\rho(w(B), w(A)) \leq s \cdot \rho(B, A),$$

从而得到 $d_H(w(A), w(B)) \leq s \cdot d_H(A, B)$.

证毕

定理 6.4.5 设 $\{w_i, i = 1, 2, \dots, N\}$ 是度量空间 X 上的一组压缩映射, s_i 为映射 w_i 的压缩因子 ($i = 1, 2, \dots, N$), 则变换

$$W: \mathcal{H}(X) \rightarrow \mathcal{H}(X), \quad A \mapsto W(A) = \bigcup_{i=1}^N w_i(A) \quad (6.4.6)$$

也是压缩的, 且压缩因子为 $s = \max\{s_i \mid 1 \leq i \leq N\}$.

证 因为有限个紧子集的并仍为紧子集, 所以对于任意 $A \in \mathcal{H}(X)$, $W(A) \in \mathcal{H}(X)$. 可见 W 确实定义了 $\mathcal{H}(X)$ 上的一个变换.

其次, 容易验证: 对于任意 $A_1, A_2, B_1, B_2 \in \mathcal{H}(X)$,

$$d_H(A_1 \cup A_2, B_1 \cup B_2) \leq \max \{d_H(A_1, B_1), d_H(A_2, B_2)\}.$$

于是, 根据定理 6.4.5, 我们有

$$\begin{aligned} d_H(W(A), W(B)) &= d_H\left(\bigcup_{i=1}^N w_i(A), \bigcup_{i=1}^N w_i(B)\right) \\ &\leq \max \{d_H(w_i(A), w_i(B)) | 1 \leq i \leq N\} \\ &\leq \max \{s_i \cdot d_H(A, B) | 1 \leq i \leq N\} \\ &= s \cdot d_H(A, B). \end{aligned} \quad \text{证毕}$$

三、迭代函数系统

一个迭代函数系统 (IFS) 是由完备度量空间 X 及其上的一组压缩变换 $\{w_i, i = 1, 2, \dots, N\}$ 构成的, 记为 $\{X; w_i, i = 1, 2, \dots, N\}$. 与该 IFS 关联的分形变换 W 定义为

$$W: \mathcal{H}(X) \rightarrow \mathcal{H}(X), \quad A \mapsto W(A) \triangleq \bigcup_{i=1}^N w_i(A). \quad (6.4.7)$$

前面已经证明, 如果变换 w_i 是压缩因子为 s_i 的压缩变换, 则上面定义的 $\mathcal{H}(X)$ 上的分形变换 W 也是压缩变换, 且压缩因子为 $s = \max \{s_i : 1 \leq i \leq N\}$:

$$d_H(W(A), W(B)) \leq s d_H(A, B), \quad A, B \in \mathcal{H}(X).$$

因此, 根据 Banach 不动点定理, 分形变换 W 有唯一不变集 A :

$$A = W(A) = \bigcup_{i=1}^N w_i(A), \quad (6.4.8)$$

而且在 Hausdorff 距离意义下,

$$A = \lim_{n \rightarrow \infty} W^n(B), \quad \forall B \in \mathcal{H}(X). \quad (6.4.9)$$

上式表明, 不变集 A 可以由变换 W 迭代作用于任意初始集 B 逼近到任意精度, 且结果与初始集无关. 因为 $(\mathcal{H}(X), d_H)$ 是完备空间, 所以 $A \in \mathcal{H}(X)$. 按动力系统的观点, 这个不变集 B 也称为 IFS 的吸引子 (attractor). 由式 (6.4.8) 知, 吸引子 A 满足下面的自拼贴 (self-tiling) 性质:

$$A = \bigcup_{i=1}^N w_i(A). \quad (6.4.10)$$

也就是说, 吸引子 A 由它在变换 w_i 下的象 $w_i(A)$ “拼贴” 而成, 或者说吸引子 A (整体) 与 $w_i(A)$ (部分) 具有自相似性^[41]. 因此, 这个吸引子具有分形结构, 或者说 A 是一个分形.

在应用中, 变换 w_i 常常选为仿射变换. 二维情形, 仿射变换具有形式:

$$w \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \quad (6.4.11)$$

仿射变换是一种应用广泛的特殊图像变换, 在分形图像编码中具有重要且基本的作用. 它是线性变换与平移变换的复合, 由六个系数完全确定. 它具有许多特殊的性质, 例如, 将平行直线映射为平行直线; 将三角形映射为三角形; 将矩形映射为平行四边形; 仿射变换的复合和逆变换 (如果存在) 仍然是仿射变换, 等等. 它能够实现一些较简单的平面映射, 常见的包括平移、旋转、缩放和剪切等等. 图 6-6 显示了一个压缩仿射变换的作用的实例.

对于吸引子来说, 由一组仿射变换确定的分形变换 W 的参数 (仿射变换 w_i 的个数 N 与系数) 称为它的分形码或 IFS 码, 它表达了吸引子部分与整体的关系.

当初 Barnsley 和他的研究小组用 IFS 模拟自然景观, 生成的与自然景物相似的具有无限细节的复杂图形都是二值图像 (黑白图像). 自然要问: 分形空间 $\mathcal{H}(X)$ 中的紧子集是如何与黑白图像联系起来的? 这涉及黑白图像的数学模型问题.

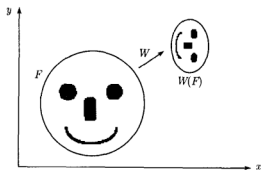


图 6-6 一个压缩仿射变换实例

在 $\mathcal{H}(X)$ 中, 基空间 (X, d) 限定为 $X \subset \mathbb{R}^2$ (例如, $[0, 1] \times [0, 1]$ 等), d 是相应的欧氏度量. 它表示了计算机屏幕或图像支持. 黑白图像在支持范围内只由黑白两色组成, 传真文件、打印文件、汉字书法等都属这一类. 在支持的白背景下, 字、图是黑色的, 如果我们把黑色赋值为 0, 白色赋值为 1, 那么支持上便有值为 1 的与值为 0 的两类集合. 假设在白背景下黑色像素的集合 S 属于 $\mathcal{H}(X)$, 即支持 X 的非空紧子集, 它完全表征了一幅支持为 X 的黑白图像. 因此, 黑白图像空间可以看成 $\mathcal{H}(X)$.

四、迭代函数系统与多功能缩印机

如何直观地理解 IFS 的概念? 多功能缩印机 (multiple reduction copying machine)^[42] 是其绝妙的比喻 (metaphor). 这台缩印机可以看成是具有下列特性的复印机:

- (1) 具有多个镜头, 可以产生原图像的多个可重叠的拷贝;
- (2) 每个镜头缩小原图像的尺寸;
- (3) 缩印机以反馈方式工作, 每次复印输出作为下一次复印的输入. 初始输入图像可以任意选择.

图 6-7 给出了一台三镜头多功能缩印机生成 Sierpinski 三角的示意图. 缩印机由三个镜头 (映射) 组成, 每个镜头把输入图像缩小一半, 并把输出图像移到如图所示的新位置. 实现缩小与平移功能的映射当然非压缩仿射变换莫属. 每个镜头的缩小功能 (映射是压缩的) 是至关重要的, 因为它保证迭代过程是收敛的. 事实上, 如果有一个镜头放大输入图像 (如放大率为 2), 不难想象, 几次反馈复印输出的图像将是一张“黑纸”.

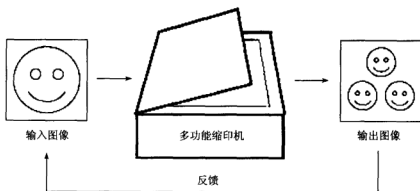


图 6-7 多功能缩印机的示意图

现在我们分别以一个内有作者姓名的长方形和黑白蝴蝶图像作为初始输入进行复印 (图 6-8): 缩印机迭代作用于两个初始图形产生两个图形序列 (从左到右, 自上而下), 结果都收敛于 Sierpinski 三角, 即这台缩印机的吸引子或不动点.

仔细分析图 6-8, 不难发现下面的事实:

- (1) 每一次复印的结果都产生更精细尺度下的细节, 如果复印无限进行下去, “最终”图像 (IFS 吸引子) 将具有无限精细的结构. 此外, 这个“最终”图像还具有不同尺度下的几何自相似性. 因此, 这个“最终”图像是分形.

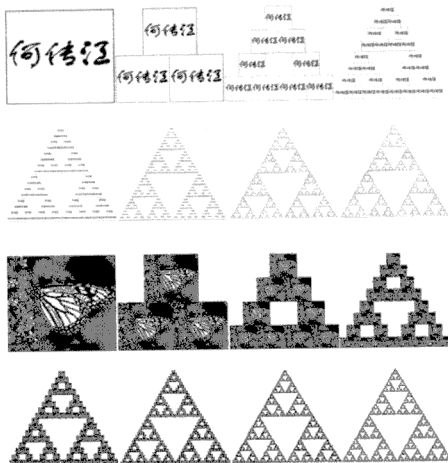


图 6-8 Sierpinski 三角构造序列

(2) 多功能缩印机几次复印的输出图像就与输入的原始图像无关了。也就是说，原始输入图像的信息差别很快消失（迭代收敛快），只要多功能缩印机功能设定（IFS 确定），“最终”的复印图像（IFS 吸引子）也就完全确定了。因此，不管原始图形是圆的、方的或人名（任意初始图像），最终都会得到 Sierpinski 三角（吸引子）。这台多功能缩印机（IFS）只会产生唯一的最终结果（吸引子唯一）。

上述多功能缩印机是一个包含有三个缩小率均为 $1/2$ 、相对位置固定的镜头的简单反馈机器，它相当于一个具有简单非线性作用规律的动力系统。如此简单的动力系统通过不断的反馈和迭代，却产生出具有复杂结构的分形图形。可见复杂的形状完全可以来自简单的动力学过程。事实上，大量的实验观测表明，长期的简单非线性动力学过程可以产生非常复杂的形状（混沌）^[43]。

多功能缩印机能够得到稳定的最终图形，是受到三个因素的控制的：多个独立

的镜头(构成 IFS 的迭代函数), 无穷次迭代过程(取极限), 反馈机器中的作用规律(IFS 变换的性质).

一个自然的问题是, 如果我们想得到一种最终的图形, 如何来设计这台多功能复印机呢? 这就提出了如何用 IFS 编码图像的问题, 我们将在下一章讨论.

第五节 本章小结

本章简单介绍了分形编码的部分数学基础, 包括度量空间、分形概念以及迭代函数系统. 要了解分形压缩编码的起源和基本原理, 这些内容是必不可少的. 本部分内容的完全介绍要涉及许多数学分支, 需要许多“深奥”的数学知识, 也将占据巨大的篇幅. 因此, 本章仅仅对这些内容作了初步介绍.

迭代函数系统理论的内容是十分丰富的, 目前仍在发展完善之中. 由于篇幅所限, 本章忽略了迭代函数系统的许多推广, 如: Local IFS^[1]、IFS with Probability^[1]、IFS with Maps^[44] 和广义自相似性^[41] 等. 此外, 本章介绍的迭代函数系统是最简单的, 没有涉及图像的亮度属性, 自相似性也仅仅限于仿射意义下. 如何推广迭代函数系统, 使得相应的分形编码适用于更广的图像类(不仅仅涉及图像的亮度属性), 这是值得进一步研究的课题. 文献 [45] 对此进行了一些探索.

参 考 文 献

- [1] Barnsley M. Fractals Everywhere. San Diego: Academic Press, CA, USA, 1988; Barnsley M, Rising III H. Fractals Everywhere (2nd ed.), Boston, MA: Academic Press Professional, 1993
- [2] Hutchinson J. Fractals and Self-Similarity. Indiana University Journal of Mathematics, 1981, 30: 713~740
- [3] 夏道行, 吴卓人, 严绍宗, 舒五昌. 实变函数与泛函分析 (下册). 北京: 人民教育出版社, 1979
- [4] 张石生. 不动点理论及应用. 重庆: 重庆出版社, 1984
- [5] Barnsley M, Hurd L. Fractal Image Compression. AK Peters, Wellesley, 1993
- [6] Barnsley M, Sloan A. A better way to compress images. BYTE, 1988, 1: 215~223
- [7] Centore P M, Vrscay E R. Continuity of attractors and invariant measures for Iterated Function Systems. Canad. Math. Bull, 1994, 37(3): 315~329
- [8] Wohlberg B, de Jager G. A Review of the Fractal Image Coding Literature. IEEE Transactions on Image Processing, 1999, 8(12): 1716~1729
- [9] Jacob F, Fisher Y, Boss R. Image compression: A study of the iterated transform method. Signal Processing, 1992, 29(12): 251~263
- [10] He C(何传江), Yang S X, Huang X(黄席樾). Progressive decoding method for fractal image compression. IEE Proc.- Vision, Image & Signal Processing, 2004, 151(3): 207~213
- [11] Qien G, Baharav Z. A new improved collage theorem with applications to multiresolution fractal image coding. 0-7803-1775-0/94, ©1994 IEEE
- [12] Domaszewicz J, Vaishampayan V A. Iterative collage coding for fractal compression. 0-8186-6950-0/94, ©1994 IEEE

- [13] Honda H, Haseyama M, Kitajima H, Matsumoto S. Extension of the collage theorem. 0-8186-8183-7/97, ©1994 IEEE
- [14] Tang Y, He C (何传江), Zhang X. A Fractal Approximation Algorithm for Inverse Initial-value Problems of Nonlinear Differential Equation. *Journal Chongqing University (Eng. End.)*, 2003, 2(2)
- [15] Mandelbrot B B, Voss R F. *Fractals: Form, Chance and Dimension*. San Francisco, CA: Freeman, 1977
- [16] Mandelbrot B B. *The Fractal Geometry of Nature* (2nd ed.). Freedman, New York, 1982
- [17] Falconer K J. *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley and Sons, 1990
- [18] Foroutan-pour K, Dutilleul P, Smith D L. Advances in the implementation of the box-counting method of fractal dimension estimation. *Applied Mathematics and Computation*, 1999, 105: 195~210
- [19] Peitgen H O, Jurgens H, Saupe D. *Chaos and Fractals: New Frontiers of Science*. New York: Springer, 1992
- [20] Pentland A P. Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984, PAMI-6: 661~674
- [21] Gharavi-Alkhansari M. Fractal-based image and video coding using matching pursuit. PhD Dissertation, University of Illinois, USA, 1997
- [22] Arduini F, Fioravanti S, Giusto D D. On computing multifractality for texture discrimination. in *Signal Processing VI: Theories and Applications, Proceedings of the Sixth European Signal Processing Conference (EUSIPCO-92)*, Aug.24~27, 1992, 1457~1460
- [23] Goel B D, Kwatra S C. A data compression algorithm for color images based on run-length coding and fractal geometry. in *IEEE International Conference on Communications'88*, June.12~15, 1988, 1253~1256
- [24] Stevens R J, Lehar A F, Preston F H. Manipulation and presentation of multidimensional image data using Peano scan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983, PAMI-5: 520~526
- [25] Keller J M, Crownover R M, Chen R Y. Characteristics of natural scenes related to the fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987, PAMI-9: 621~627
- [26] Peleg S, Naor J, Hartley R, Avnir D. Multiple resolution texture analysis and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984, PAMI-6: 518~523
- [27] Peli T. Multiscale fractal theory and object characterization. *Journal of Optical Society of America A*, 1990, 7: 1101~1112
- [28] Theiler J. Estimating fractal dimension. *Journal of Optical Society of America. A, Optics and Image Science*, 1990, 7: 1055~1073
- [29] Moghaddam B, Hintz K J, Stewart C V. A comparison of local fractal dimension estimation methods. *Pattern Recognition and Image Analysis*, 1992, 2: 93~96
- [30] Jang J, Rajala S. Segmentation based image coding using fractals and the human visual system. in *Proceedings of IEEE ICASSP-90*, Albuquerque, NM, Apr.3~6, 1990, 1957~1960
- [31] Jang J, Rajala S. Texture segmentation-based image coder incorporating properties of the human visual system. in *Proceedings of IEEE ICASSP-91*, Toronto, Canada, May 14~17, 1991, 2753~2756

- [32] Chaudhuri B B, Sarker N. Texture segmentation using fractal dimension. *IEEE Transaction on PAMI*, 1995, 17(1): 72~77
- [33] Keller J M, Chen S. Texture description and segmentation through fractal geometry. *Computer Vision, Graphic and Image Processing*, 1989, 45: 150~166
- [34] Keller J M, Chen S. Texture description and segmentation through fractal geometry. *Computer Vision, Graphic and Image Processing*, 1989, 45:150~166
- [35] Sarker N, Chaudhuri B B. An efficient approach to estimate fractal dimension of texture image. *Pattern Recognition*, 1992, 25: 1035~1041
- [36] Cheong C K, Aizawa K, Saito T, Hatori M. Structural edge detection based on fractal analysis for image compression. in *IEEE International Symposium on Circuits and Systems*, San Diego, CA, May10~13, 1992, 2461~2464
- [37] Barnsley M F, Jacquin A E. Application of recurrent iterated function systems to images. in *Proceedings of the SPIE, Visual Communications and Image Processing*, 1988, 1001: 122~131
- [38] Jacobs E W, Boss R D, Fisher Y. Fractal-based image compression II. Naval Ocean Systems Center, San Diego, CA, Tech. Rep. 1362, June 1990
- [39] Jacquin A E. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1992, 1(1): 18~30
- [40] 潘金贵等编著. 分形艺术程序设计. 南京: 南京大学出版社, 1998
- [41] Cabrelli C, Molter U. Generalized self-similarity, *Journal of Mathematical Analysis and Applications*. 1999, 230: 251~260
- [42] Peitgen H O, Jurgens M, Saupe D. Encoding images by simple transformations. *SIGGRAPH '90 Course Notes, Fractals: Analysis and Modeling*, vol. 15, chapter 11, 1990, pp 1~21
- [43] 王东升, 曹磊. 混沌、分形及其应用. 合肥: 中国科学技术大学出版社, 1995
- [44] Forte B, Vrsay E R. Solving the inverse problem for functions and image approximation using iterated function systems. *Dyn. Cont. Impul. Sys.*, 1995, 1: 177~231
- [45] Cabrelli C, Falsetti M, Molter U. Fractal Block-Coding: A Functional Approach for Image and Signal Processing. *Computers and mathematics with Applications*, 2002, 44: 1183~1200

第七章 基本分形编码算法

第一节 引言

分形图像编码是一个相对较新的图像压缩技术. 在分形图像编码中, 通过利用自然图像中存在的不同子图像间的跨尺度相似性(即把图像视为分形), 一幅图像用一个使图像近似不变的压缩仿射变换^①的参数来表达, 压缩文件中储存的是这些参数的量化值而不是图像本身的像素值. 显然, 存储这些量化参数的比特数大大低于存储原始图像的比特数, 因而实现了图像数据的高倍压缩. 解码是新颖的快速迭代过程, 原始图像 μ_{org} 由变换的不动点来重构: $\mu_{\text{org}} \approx \mu_T = \lim_{n \rightarrow \infty} T^n(\mu_0)$, 式中 μ_0 是任意初始图像, T 是表达图像的压缩变换. 因为解码是迭代过程, 每次迭代都会产生更精细尺度下的细节, 也就是说, 解码图像(从理论上)具有任意分辨率下的细节. 自相似结构和具有任意分辨率下的细节都是分形的典型特征, 因此, 解码图像具有分形的特征. 目前, 分形图像编码以其新颖的思想、高压缩比、分辨率无关性等优点受到技术界广泛关注, 是公认的三种最有前途的新一代图像编码技术之一(另两个是小波技术与模型法技术)^[1].

分形图像编码是 Barnsley 于 1988 年提出的美国专利技术, 源于他们对迭代函数系统的研究, 该技术对几幅图像的分形编码获得了难以置信的超高压缩比(10000 : 1)^[2]. Barnsley 在文献 [3] 中描述的分形编码大意如下: 通过一些如颜色分割、边缘检测、频谱分析、纹理分析等图像处理技术, 从图像中提取分形子图(如一棵树、一片云、一片海景), 每个子图具有一定的分形结构——子图的整体与局部之间存在某种自仿射特征, 由此构造许多不同形态子图组成的分形库(当然不是以图像格式存入的, 否则库的容量将极其巨大, 有悖于压缩的初衷). 分形库中每个子图由一个迭代函数来表征, 这些迭代函数具有仿射形式, 故只要几个参数即可确定. 也就是说, 分形库存储的并不是子图本身, 而是由若干参数组成的 IFS 码(仿射迭代函数参数), 因此, 分形库并不占有很大的储存空间. 编码阶段, 一幅图像按某种方式划分成互不重叠、形状可能任意的子图. 对于每个子图, 在分形库中寻找与之最仿射相似的子图(即在旋转、反射、伸缩、偏斜等操作之后与之相似, 这一过程需要人机交互), 以保证它们“拼贴”(collage)起来与原始图像相似. 于是, 一幅图像对应于一个由若干迭代函数组成的迭代函数系统, 迭代函数系统的所有参数

^① 这个压缩仿射变换是由一组刻画图像局部自相似性的压缩仿射函数组成的, 通常称为迭代函数系统(iterated function system).

就构成图像的 IFS 码。因此, 这种编码方法对具有典型分形结构的图像能够实现非常高的压缩比^[2]。

但上述方法是不实用的, 因为编码时间太长, 且需要人机交互, 对操作者有较高要求。尽管如此, 它毕竟为图像压缩提供了一条与以往完全不同的新思路。实际上, Barnsley 的博士生 Jacquin 提出的实用分形块编码^[4]正是在此基础上发展起来的。此外, 上述方法较好地利用了人眼视觉系统的特性, 比较符合人的视觉感知规律。例如, 人们对一幅图像的初步感知并不会注重一些特别的细节, 而是该图像中包含的一些具有特征意义的子图, 因为人的视觉感知是一种宏观认识过程。从这个意义上说, 分形编码类似于模型编码^①。最后, 把上述方法与 CAD 技术对比, 我们会得到不少启示。在 CAD 技术中, 无论生成的图形多么复杂, 但存储该图形的文件数据量是相当少的, 这是因为我们只要知道构成图形的点、直线、圆等基本元素以及构成的过程就可以了。自然图像之所以不能如此, 一个重要的原因就是对那些广泛存在的自然结构不能用欧氏几何来描述。现在有了分形理论的指导, 在形式上对图像表达进行简化就有了可能, 正如 Barnsley 型分形编码方法实现的那样。

在分形编码实用化的研究中, Jacquin 迈出了实质性的一步, 在其著名论文^[4]里提出了分形块编码 (不妨称为 Jacquin 型编码), 这是一种基于方块划分的计算机自动分形编码算法。分形块编码算法建立在图像局部自相似的基础上, 在最简情形, 图像被分割成大小两类子块, 小图像互不重叠且覆盖整幅图像。然后对于每个小方块, 在图像中搜索与之最仿射相似的大方块, 这样每个小方块由与其匹配的大方块的仿射变换 (空域收缩、旋转、反射与亮度变换) 来近似。这些作用于大块的变换以分块方式定义一个作用于整幅图像上的分形变换, 其不动点逼近待编码图像。这给出了图像数据的一种自相似描述, 解码后的图像呈现出一定的自相似结构。本书作者认为, Jacquin 型编码把分形编码从“专家算法”变成“大众算法”^②。事实上, 正是 Jacquin 型编码算法出现之后, 分形图像编码才真正成为一个多领域学者参与的活跃研究领域。本书仅仅讨论这种分形编码, 因此, 若无特别说明, 以后凡是提到分形编码, 都是指 Jacquin 型编码。

图像的分形编码是一种利用块间自相似性来减少图像数据冗余度的新型编码技术, 它具有以下特点:

(1) 思想新颖。分形编码是利用自然图像中不同区域间存在的跨尺度冗余 (即不同尺度下的自相似冗余) 来实现图像压缩的。自然图像是大自然某个部分的真实再现, 而自然界的景物图像都存在确定的或统计的自相似性, 分形编码算法恰好利用了这一点。

(2) 压缩比很高。对一般图像, 当压缩比在 20 倍以上时, 仍有较好的保真度。

① 作者认为, 把这种编码方法称为“分形模型编码”能够较好反映其本质特征。

② 弄懂 Barnsley 的“分形模型编码”的原理需要坚实的数学基础, 此外, 人机交互式的编码对操作者又有较高要求, 因此它是一种“专家算法”, “分形块编码”却没有这些“缺点”。

对某些自相似性强的图像, 压缩比可达上百倍. 研究表明, 与基于 DCT 的 JPEG 编码相比, 当压缩比在 40 倍以上时, 分形编码能取得更好的图像质量; 但在较低的压缩比下, JPEG 编码的效果更好. 因此, 分形图像编码可以作为 JPEG 编码的有益补充.

(3) 解码图像的分辨率无关性. 可按任意高于或低于原编码图像的分辨率来进行解码, 当要解码成较高分辨率图像时, 引入的细节会与整个图像大致和谐一致, 从而比像素复制或插值方法得到的图像看起来更自然. 这种缩放能力也可以用作图像增强工具.

(4) 解码速度快. 分形压缩是一非对称过程, 虽然基本分形编码很耗时, 但解码速度快 (且是新颖的迭代解码方式), 因此, 分形编码为图像存储与恢复等一次编码多次解码的应用场合提供了一个优秀的候选方法——编码可由专用硬件设备完成, 解码则可按用户的需要用软件方法多次重复进行.

(5) 编码时间过长, 实时性差. 但是, 目前已提出了大量快速分形算法, 分形编码的时间已经大大缩短.

分形编码的研究虽只有十余年的历史, 但已成为最有发展前途的图像编码方法之一, 也是一种研究范围较为广泛的编码方法. 虽然目前大多数纯粹基于分形的编码技术与目前广泛使用的压缩技术相比, 还缺乏竞争力, 但是, 我们不能以过分挑剔的态度对待这一新兴的技术. 实际上, 分形压缩与其他编码技术结合的混合编码技术已经取得巨大成功^[6]. 此外, 作为分形图像压缩技术的应用, 一个成功例子是“Microsoft Encarta”光盘^[5]. 该光盘是微软公司推出的一本多媒体百科全书, 广泛收集了文章、动画片、声音、插图、照片、地图册和字典, 内有几千幅彩色地图 (可局部放大), 几千张优质彩色照片 (可交互式查看). 这么多内容, 就是使用分形图像压缩技术全部压缩存储在一张光盘里 (现已扩展为三张光盘).

任何新方法提出后, 从不同的观点对它加以研究总结总是一件十分有意义的工作. 不同的观点一方面能够使我们更好地了解方法的本质, 另一方面也能够启发我们进行新的研究, 分形图像压缩方法自然不能例外. 事实上, 自分形图像编码出现以来, 人们已从多种不同的观点对它进行了深入研究^[7].

本章拟从迭代函数系统观点介绍基本分形编码的基本原理, 从矢量量化观点介绍其算法及实现. 这样做是出于这样的考虑: 迭代函数系统是分形图像编码的起源, 是分形图像编码的传统方法, 也是学习分形编码最好的教材. 但对于分形图像压缩软件的设计, 矢量量化观点更适合、更直接, 因为它本身就是直接处理离散对象的.

第二节 矢量量化与分形编码

我们将看到, 尽管分形编码起源于迭代函数系统 (iterated function systems,

IFS),但是,它与矢量量化编码(vector quantization, VQ)十分类似.在编码阶段,两种方法都要使用一个称为码本(codebook)的向量集来近似原始图像的每个子块.在VQ编码中,码本生成于一组训练图像,并用于编码任何图像(未必是训练图像).但为了重构原图像,解码器也必须有一本同样的码本.在分形图像编码中,码本源于原始图像,而且解码器并不存在也不需要这本码本,因为分形解码是迭代解码,解码器只需按照分形文件描述的压缩变换迭代作用于任何同尺寸图像即可恢复原图像.

本节简单介绍矢量量化的基本思想,然后从矢量量化的观点自然引出分形编码.这样做有助于具有图像编码知识的工程技术人员更好地理解分形编码,不至于一开始就被“深奥”的数学理论所难住.当然,要完全理解分形编码的内涵,还得从迭代函数系统入手.我们将在后两节中讨论这些内容.

一、矢量量化

矢量量化(VQ)是20世纪80年代发展起来的编码技术,也是一个发展比较成熟、应用十分广泛的编码技术^[8~10].在语音与图像编码方面,VQ目前已显示出不少优越性.VQ编码器与声码器相结合的编码,能以150 bps(bits per second)的数码率得到易懂的语音;VQ对图像的编码,将数码率压低到0.7bpp(bits per pixel,即图像编码的数码率,一般用每像素的平均二进制符号个数表示),仍然可以得到失真不大的图像^[8].这些都是它以前的编码方法不易办到的^①.

许多问题都涉及VQ问题,例如,考虑在仅仅支持256色颜色查找表的图形卡上显示真彩色图像,就必须使用VQ方法.实际上,颜色查找表的最优设计已成为计算机图形学的一个基本问题^[11].

矢量量化实质上是把一维量化(数值量化、标量量化)推广成多维量化,VQ编码器的输入 x 是 n 个实数构成的向量.为了送进计算机,这 n 个实数一般是用二进制写成的,它们可以是图像的灰度、语音的样本值等.

矢量量化包括编码和解码两部分.编码器和解码器各有一本同样的码本: $Y = \{y_1, y_2, \dots, y_j, \dots, y_N\}$,码本 Y 的元素 y_j 叫做码字(codeword),也叫代表向量.当输入向量 x 时,编码器将 x 与码本中的各码字比较,看它和哪一个码字最接近,比方说是 y_j ,因为接收端也有一本同样的码本,编码器只要输出码字 y_j 的索引号(index) j 就行了.解码器可以根据索引号 j 从码本中查出 y_j .因此,存储或传输的仅仅是索引号而不是向量本身,从而实现了数据的压缩.

不难看出,VQ编码有如下的特点^[8]:

① 对VQ有兴趣的读者,可以阅读Gray和Neuhoff的特邀综述文章^[10].该文介绍了自1948年(三篇著名的奠基性论文发表年)以来量化理论与应用的发展情况,所列581篇参考文献基本上包括了量化方面的主要内容.

(1) 一定产生失真. 如果任意一个 x , 码本都有和它一样的码字 y , 即 $y = x$, 那么 VQ 是无法压缩数据的.

(2) 压缩能力强. 编码实际上是将大体上相似的向量分为一类, 用一个代表向量代替这一类. 假如我们把 100 种不同的向量归为一类, 压缩比就可以达到 100:1. 当然, 实际并非如此简单地归类, 这里只是想说明它的概念而已.

(3) 失真量易控制. 向量的分类越细, 失真就越小. 这意味着码本容量越大, 失真越小. 只要适当选择码字数量, 就能控制失真量不超过某一给定值.

(4) 计算量大. 每输入一个向量 x , 我们都要将它与码本中的每个码字 y 逐一进行比较, 看它和谁最相似 (全搜索). 因为 x 和 y 都是向量, 所以全搜索的运算是向量运算, 工作量很大, 这是 VQ 的一个显著缺点.

VQ 编码的首要问题是如何以最优方式设计码本, 使得上面描述的量化过程平均而言产生最小的失真.

为了解决码本设计问题, 必须首先定义量化失真程度的函数, 这个函数在量化理论中称为失真测度 (distortion measure). 大多数情况下, 失真测度取为欧氏距离的平方:

$$d(x, y) = \|x - y\|^2 = \sum_{k=1}^n (x_k - y_k)^2, \quad x, y \in R^n, \quad (7.2.1)$$

其中, n 表示量化器的维数 (在颜色量化的例子中, $n=3$), x_i 是向量 x 的第 i 个分量. 用这种失真测度的量化器称为最近邻量化器 (nearest-neighbor quantizer)^[9].

码本的最优设计是非常困难的问题, 实际应用中往往只能得到次优解. 下面我们介绍一种目前使用最普遍的方法, 叫作随机量化.

假设待设计的码本为 $Y = \{y_1, y_2, \dots, y_N\}$, 失真测度由式 (7.2.1) 定义. 随机收集源于训练图像的 M 个 n 维数据向量 $x_1, x_2, \dots, x_M \in R^n$, 这个序列叫作向量训练序列 (training sequence). 其中 M 应该远大于 N , 从而使训练序列的概率分布具有一定的统计代表性. 在这 M 个向量中, 有些是很靠近甚至是相同的, 我们可以根据密集程度把它们划分成 N 个集合 R_1, R_2, \dots, R_N , 越是密集的部分, 划分越细, 这样可以减小失真. 当然, R_1, R_2, \dots, R_N 应该将 M 个训练向量 x_1, x_2, \dots, x_M 无遗漏地划分成 N 个互不相交的集合, 即划分应该满足

$$\{x_1, x_2, \dots, x_M\} = \bigcup_{i=1}^N R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j \quad (7.2.2)$$

这就意味着任一训练向量必定属于 R_1, R_2, \dots, R_N 中的一个而且仅仅一个.

其实我们现在是以训练序列 x_1, x_2, \dots, x_M 代替 n 维向量空间 R^n , 以训练向量集合 $\{x_1, x_2, \dots, x_M\}$ 的 N 个子集合 R_1, R_2, \dots, R_N 代替 R^n 的 N 个区域 R_1, R_2, \dots, R_N 来进行矢量量化. 如果下列两个最优性条件满足, 我们就说量化是最优的, 至少对训练序列来说是最优的.

(1) 最近邻条件 (Nearest Neighbor Condition) 给定一本码本 $Y = \{y_1, y_2, \dots, y_N\}$, 最优划分 R_i 必须满足:

$$R_i \subset \{x \in R^n \mid d(x, y_i) \leq d(x, y_j), \quad \forall j = 1, 2, \dots, N\}.$$

因此, 对于给定向量 x 的量化, 失真量等于 $d(x, Q(x)) = \min_{y_j \in Y} d(x, y_j)$.

(2) 形心条件 (Centroid Condition) 给定划分 R_1, R_2, \dots, R_N , 最优码本 $Y = \{y_1, y_2, \dots, y_N\}$ 由向量集 R_i 的形心组成:

$$y_i = \text{centroid}(R_i) \triangleq \left(\sum_{j=1}^M 1_{R_i}(x_j) x_j \right) / \sum_{j=1}^M 1_{R_i}(x_j), \quad (7.2.3)$$

其中, $1_R(x)$ 表示指示函数: $1_R(x) = 1, x \in R; 1_R(x) = 0, x \notin R$. 由于划分满足式 (7.2.2), 所以 $\{x_1, x_2, \dots, x_M\}$ 中的任一向量 x , 总会使 $1_{R_1}(x), 1_{R_2}(x), \dots, 1_{R_N}(x)$ 中有一个且只有一个为 1, 其余为 0. 于是, $\sum_{j=1}^M 1_{R_i}(x_j) x_j$, $\sum_{j=1}^M 1_{R_i}(x_j)$ 分别是属于 R_i 的向量 x_j 的和与向量个数, 式 (7.2.3) 计算的 y_i 就是属于 R_i 的向量 x_j 的平均向量 (向量系统的形心).

不难看出, 最近邻条件实际上给出了训练向量集 $\{x_1, x_2, \dots, x_M\}$ 的一种划分, 根据码本将训练向量集 $\{x_1, x_2, \dots, x_M\}$ 划分成若干类, 即把与码字 y_i 最相似的 x_j 归为同一类 R_i . 形心条件则给出了依照划分确定更新码本的方法, 即码本由诸 R_i 的形心组成. 因此, 在上述两个条件下, 码本的设计等价于向量集 $\{x_1, x_2, \dots, x_M\}$ 的划分. 下面的迭代过程就是这个事实的反复交替使用.

现在我们给出最优码本的一种设计算法——LBG 算法^[12]. 给定训练向量序列 $\{x_1, x_2, \dots, x_M\}$ 和码本 Y_m . 根据上述两个最优性条件, 从码本 Y_m 出发, 一个改进的码本 Y_{m+1} 按广义 Lloyd 迭代生成:

Step 1 给定一本码本 $Y_m = \{y_1^{(m)}, y_2^{(m)}, \dots, y_N^{(m)}\}$, 按下面的最近邻条件把训练集 $\{x_1, x_2, \dots, x_M\} \subset R^n$ 划分为 N 个子集 $R_i^{(m)}$:

$$R_i^{(m)} = \{x_k \mid d(x_k, y_i^{(m)}) \leq d(x_k, y_j^{(m)}), \forall j\}, \quad 1 \leq i \leq N, \quad (7.2.4)$$

其中需要适当的平局决胜规则 (tie-breaking rule). 例如, 如果 $d(x_k, y_j) = d(x_k, y_i)$, x_k 是归入 R_j 还是 R_i , 总得确定一个规则.

Step 2 使用形心条件计算子集 $R_i^{(m)}$ 的形心, 并更新码本如下:

$$Y_{m+1} = \{y_i^{(m+1)} \mid y_i^{(m+1)} = \text{centroid}(R_i^{(m)}), i = 1, 2, \dots, N\}. \quad (7.2.5)$$

上述过程是迭代进行的: 给定初始码本 $Y_0 = \{y_j^{(0)}, 1 \leq j \leq N\}$, 按步骤 1 得到训练集的一种划分 $\{x_i, 1 \leq i \leq M\} = \bigcup_{i=1}^N R_i^{(0)}$, 步骤 2 计算每一类 $R_i^{(0)}$ 的形心, 它

们的集合就构成更新的码本 $Y_1 = \{\mathbf{y}_j^{(1)}, 1 \leq j \leq N\}$; 按照更新码本 Y_1 , 步骤 1 得到训练集的又一种划分 $\{\mathbf{x}_i, 1 \leq i \leq M\} = \bigcup_{i=1}^N R_i^{(1)}$, 步骤 2 计算每一类 $R_i^{(1)}$ 的形心 $\mathbf{y}_i^{(1)}$, 它们的集合就构成更新的码本 $Y_2 = \{\mathbf{y}_j^{(2)}, 1 \leq j \leq N\}$; 反复进行这个过程直到满足终止判据为止。

不难得到, 从初始码本 Y_0 连同初始量化器 Q_0 出发, 第 m 次迭代后的总失真量等于

$$D_m = \sum_{k=1}^M d(\mathbf{x}_k, Q_m(\mathbf{x}_k)) = \sum_{k=1}^M \min_{\mathbf{y}_j \in Y_m} d(\mathbf{x}_k, \mathbf{y}_j^{(m)}). \quad (7.2.6)$$

根据最优性条件, 总失真序列 $\{D_m\}$ 是单减的, 又是有下界的 (D_m 显然是非负的), 因此根据微积分学的一个基本定理——“单调有界数列必收敛”, 序列 $\{D_m\}$ 是收敛的。此外, 有限训练向量集只有有限种划分方式, 因此, 序列 $\{D_m\}$ 的极限必在有限次迭代后达到。在实际操作中, 一般使用下面的终止判据:

$$|D_{m+1} - D_m|/D_m = (D_m - D_{m+1})/D_m \leq \varepsilon, \quad (7.2.7)$$

其中, $\varepsilon > 0$ 是用户指定的容忍误差。

上面就一般信源讨论了 VQ 编码, 下面我们回到图像的 VQ 编码问题。从理论上说, 只要扩大码本的容量, 标准 VQ 编码就可实现好的率失真曲线 (rate-distortion curve)^①, 而且图像中的边缘等重要特征也能被很好保持。但是, 仅仅靠增大码本来实现好的率失真曲线是不切实际的, 原因有二: 一是存储容量需求太大, 因为编码器和解码器都要存储一本这样的码本; 二是码本设计算法因海量时间花费而无法实现。例如, 假设待编码图像为 8 bit 量化的 512×512 图像, 用尺寸为 $d \times d$ 的方块划分之, 若以压缩率 CR 计, 则码本容量应为 $2^{8n/CR} (n = d^2)$ 。若以 $CR = 10$ 和 $d = 8$ 计, 码本的码字数是十分惊人的。

因为这个原因, 人们提出了许多不同的 VQ 编码方案^[10], 主要是赋予码本一定的结构以使计算可行 (寻求次优解), 当然这是以牺牲一定的图像质量为代价的。乘积码 VQ (product code VQ)^[10] 就是一个著名方法, 这里介绍其特殊情形 MSG-VQ (Mean-Shape-Gain VQ), 它是 Murakami 等人^[13] 首次引入的。在 MSG-VQ 中, 给定一个向量 $\mathbf{R} = (r_1, r_2, \dots, r_n) \in R^n$, 定义其均值 o 、增益 (gain) s 和形态向量 (shape vector) 如下:

$$o = \frac{1}{n} \sum_{i=1}^n r_i, \quad s = \left(\sum_{i=1}^n |r_i - o|^2 \right)^{1/2}, \quad D = \frac{1}{s} (\mathbf{R} - o \cdot \mathbf{1}), \quad (7.2.8)$$

① 一个可变率编码器 (variable rate encoder) 能够实现不同的压缩率, 从而得到不同质量的编码。因此, 为了对不同的编码器或同一个编码器的不同参数设置进行比较, 我们需要为两种情况在坐标系上记录一些点 (坐标系的横轴为压缩比, 纵轴为 PSNR), 连接这些点画出的曲线称为率失真曲线。

于是向量 R 能够分解成

$$R = s \cdot D + o \cdot 1, \quad (7.2.9)$$

其中, $1 = (1, 1, \dots, 1)^T \in R^n$. 显然 $D = (d_1, d_2, \dots, d_n) \in R^n$ 是均值为零、方差为 1 的向量, 即 $\sum_i d_i = 0, \sum_i d_i^2 = 1$. 向量 R 的 MSG-VQ 编码, 就是用失真误差

$$d(R, \hat{R}) = \sum_{i=1}^n |r_i - \hat{r}_i|^2 \text{ 最小的重构向量 } \hat{R} = \hat{s} \cdot \hat{D} + \hat{o} \cdot 1 \text{ 近似它.}$$

给定参数 s, o 的标量码本和形态向量 D 的向量码本 (共三本码本), 输入向量 R 的量化形式为

$$R \approx s_{\text{ind}_s(R)} \cdot D_{\text{ind}_D(R)} + o_{\text{ind}_o(R)} \cdot 1 \quad (7.2.10)$$

其中 $\text{ind}_s(R)$ 、 $\text{ind}_o(R)$ 和 $\text{ind}_D(R)$ 分别是量化器生成的三本相应的码本索引号 (自然涉及如何确定这些索引号的问题, 最基本的方法就是全搜索). 由式 (7.2.9) 知道, o 与 s 分别是 R 的均值与方差, 一种次优方案就是独立地编码向量 R 的均值、方差和形态向量, 因为同时考虑三本码本将产生海量联合码本 (使编码时间太长以致不可行). 例如, 假设 s, o 分别用 5 bit、7 bit 量化 (s, o 的码本大小分别是 32、128), 并设形态码本由 $2^{12} = 4096$ 个码字组成 (12 bit 量化), 则容易计算出联合码本 $C_o \times C_s \times C_v$ 的大小为 2^{24} . 其中, C_o, C_s 和 C_v 分别是 s, o 和形态向量的码本.

关于 VQ 及各种变形可以在文献 [9,10] 中找到详细资料, 我们不再介绍了.

二、从矢量化观点看分形编码

在 MSG-VQ 中, 图像块由 DC 分量与取自形态码本 C_v 的图像块的亮度调整之和来近似, 即式 (7.2.9), 其中码本 C_v 是经训练得到的、与待编码图像无关的“固定码本”, 也就是说, 一旦设计好这样的码本, VQ 编码器对所有图像都采用相同的码本来编码. 一个自然的想法是考虑用“自适应码本”代替固定码本, 即 VQ 编码器采用适应于图像的码本, 对不同的图像用不同的码本来编码. 要实现这个想法, 最起码的要求, 这样的“自适应”VQ 编码器在解码阶段不需要这样的码本.

读者也许会认为这蕴涵矛盾, 恢复原图像正是解码器的工作, 因此解码器自然不能访问原图像, 从而不能得到重构图像所需的“自适应码本”, 图像重构如何进行呢?

为此, 我们举一个简单的例子以阐明上述想法是可以实现的. 为简单起见, 假设待编码的是一个数 (如自然对数的底 $e = 2.718\dots$), 而不是一幅图像 (也可看成是 1×1 图像). 并设 s, o 的码本如下:

$$C_s = \{0.0, 0.25, 0.5, 0.75\}, \quad C_o = \{0.0, 0.4, 0.8, 1.2, 1.6, 2.0\}. \quad (7.2.11)$$

此时, 形态向量码本 C_v 由一个数组成 (为简单计, 假设为 e 本身). 表 7-1 给出了所有可能的 $s \cdot e + o$, 其中 s, o 分别取自各自的码本.

表 7-1 所有可能的 $s \cdot e + o$ (截断到两位小数), s, o 分别源于各自码本

Scale s	Offset o					
	0.00	0.40	0.80	1.20	1.60	2.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.25	0.68	1.08	1.48	1.88	2.28	2.68
0.50	1.36	1.76	2.14	2.54	2.94	3.34
0.75	2.04	2.44	2.84	3.24	3.64	4.04

表 7-1 列出的所有可能的数 $s \cdot e + o \cdot 1$ 的集合, 相当于 MSG-VQ 中的乘积码本. 对乘积码本进行全搜索, 得到 $s = 0.25, o = 2.0$ 给出了 e 的最好近似, 即

$$s \cdot e + o = 0.25 \cdot e + 2.0 = 2.6795 \cdots \approx e.$$

因此, 按 MSG-VQ 编码, 只需把 s, o 的索引号 $\text{ind}_s(e) = 2, \text{ind}_o(e) = 6$ 传送给解码器, 然后解码器根据这两个索引号从码本 C_s, C_o 中查找出 $s = 0.25, o = 2.0$, 最后按 $s \cdot e + o \cdot 1 = 0.25 \cdot e + 2.0$ 恢复出 e 的近似值 2.6795. 显然, 这种编码方法需要解码端也有三本同样的码本. 这是按乘积 VQ 编码方法进行的.

我们也可以按另一种方式进行. 编码器可以传输下面的信息给解码器:

“所传的数约等于自己的 0.25 倍加 2.0”.

因为没有指定误差, 有很多数都满足这个条件, 所以, 如果没有其他信息, 解码器可以确定其中的任何一个数作为解码输出. 然而, 其中有一个数是唯一确定的, 即“所传的数准确等于自己的 0.25 倍加 2.0”:

$$x = 0.25x + 2.0, \quad (7.2.12)$$

解这个简单方程得到 $x \approx 2.6$, 它将作为解码输出 (e 的近似值).

方程 (7.2.12) 是容易求解的, 但是, 我们要编码的图像绝不会是 1×1 的 (一个数), 而是一个很大的阵列, 如 512×512 , 因此类似于 (7.2.12) 的方程是一个很大的方程组, 直接计算显然是不可能的, 必须寻求其他方法.

再次回到简单方程 (7.2.12), 如果我们定义一个算子

$$T: \mathbb{R} \rightarrow \mathbb{R}, \quad Tx = 0.25x + 2.0, \quad (7.2.13)$$

那么编码器传给解码器的信息简单表示为 $e \approx Te$, 即 e 是算子 T 的近似不动点 (当然需要把算子 T 的描述——“把一个数变成自己的 0.25 倍加 2.0”——传给解码器). 因此, 解码器只需求解不动点方程 $Tx = x$. 为此, 不难联想到 Banach 不动点定理.

实数集 \mathbb{R} 可以看成是赋予了绝对值度量的度量空间, 且是完备的, 算子 T 显然是完备度量空间 \mathbb{R} 上压缩因子为 0.75 的压缩变换. Banach 不动点定理保证 T 的不动点是唯一存在的, 且可以从任何数 x_0 出发迭代生成. 例如, 设 $x_0 = 0$, 我们给出前几次迭代的结果如下:

$$x_1 = Tx_0 = 2.0, \quad x_2 = Tx_1 = 0.25 \times 2.0 + 2.0 = 2.5,$$

$$x_3 = Tx_2 = 0.25 \times 2.5 + 2.0 = 2.625,$$

$$x_4 = Tx_3 = 0.25 \times 2.625 + 2.00 = 2.65625,$$

$$x_5 = Tx_4 = 0.25 \times 2.65625 + 2.0 = 2.6640625,$$

这个数列 $\{x_n\}$ 将收敛于不动点 2.66... 选取其他的数作为初始数, 重复上述迭代过程仍然可以得到同样的结果.

从上面例子可以看出, 按照后一种编码方法, 编码器只要描述一个式 (7.2.13) 那样的算子 T , 解码器只要根据 T 的描述 (e 的码) 恢复出算子 T 来, 然后迭代作用于任意的初始数 (初始数影响迭代收敛速度), 一定的迭代次数后就得到待编码数 e 的近似值. 整个解码过程并不需要待编码数 e 的任何信息, 需要的仅仅是算子 T 的描述.

同样的思想应用于图像编码, 就产生分形编码方法. 举个例子说, 假设待编码的是 512×512 图像, 被分割为 8×8 的不重叠方块 (R 块), 然后从原图像中选取 16×16 块 (D 块), 并按像素值平均得到 8×8 块. 所有这样的 8×8 块 D 构成“自适应码本”(相当于 MSG-VQ 中的形态向量码本, 但并不需要传递给解码器). 其次, 按某种方式事先构造 s, o 的码本.

编码阶段, 对于每个 R 块 R , 按某种方法寻找其最佳近似: $s \cdot D + o \cdot 1 \approx R$. 这样得到的 s, o 和 D 的索引号就是 R 的分形码, 它描述了 R 是如何近似得到的. 全体 R 的分形码就构成待编码图像的分形码, 它描述了作用于整幅图像的算子 (要保证压缩性, 类似于式 (7.2.13)). 解码阶段, 解码器只要根据 T 的描述 (分形码) 恢复出算子 T 来, 然后迭代作用于任意的初始图像, 一定的迭代次数后就得到待编码图像的近似. 容易看出, 整个解码过程并不需要待编码图像的任何信息, 需要的仅仅是算子 T 的描述 (这是需要储存或传输的).

上面没有说明, 对于每个 R 块 R , 如何寻找其最佳近似 $s \cdot D + o \cdot 1$. 我们当然可以按编码 e 的方式事先构造 s, o 的码本和“自适应码本”, 然后从这三本码本的联合码本中确定 R 的最佳近似: $R \approx s \cdot D + o \cdot 1$, 即搜索出系数 s, o 和 D 在各自码本中的索引号. 但实际有时无法实现, 因为 $s \cdot D + o \cdot 1$ 是很多的, 计算量太大, 编码时间太长. 一种可行方法是先确定系数 s 和 o (如用最小二乘法), 然后搜索“自适应码本”并从中找出最佳的 D .

但是, 要了解分形编码的起源和数学原理, 我们必须从迭代函数系统的观点介

绍分形编码. 这就是下面两节的主要内容.

第三节 迭代函数系统正问题与自然图形模拟

为什么基于迭代函数系统 (IFS) 的分形方法能压缩图像? 要讲清楚这个问题, 必须从自然图形的 IFS 建模谈起. 分形技术引入计算机图形学时就主要用来模拟自然景观. 20 世纪 80 年代后期, Barnsley 和他的研究小组发现用简单的 IFS 编码就可生成与自然景物相似的具有无限细节的复杂图形, 由此看到 IFS 建模某些具有分形特征的自然景观 (如云彩、树叶、山脉、海岸线等) 的潜力, 并用 IFS 逼真地再现了这些自然分形图形^[14,15].

为阅读方便, 下面简要回顾一下迭代函数系统. 一个迭代函数系统是由完备度量空间 X 及其上的一组压缩仿射映射构成的, 即 $\{w_i; i = 1, 2, \dots, N\}$, 与这个 IFS 关联的分形变换 W 定义为

$$W: \mathcal{H}(X) \rightarrow \mathcal{H}(X), A \mapsto W(A) = \bigcup_{i=1}^N w_i(A). \quad (7.3.1)$$

我们已经知道, 如果压缩变换 w_i 的压缩因子为 s_i , 则上面定义的变换 W 也是压缩变换, 且压缩因子为 $s = \max\{s_i; 1 \leq i \leq N\}$:

$$d_H(W(A), W(B)) \leq s d_H(A, B), \quad \forall A, B \in \mathcal{H}(X)$$

因此, 根据 Banach 不动点定理, 分形变换 W 有唯一不变集 (吸引子):

$$A = W(A) = \bigcup_{i=1}^N w_i(A), \quad (7.3.2)$$

而且在 Hausdorff 距离意义下, $A = \lim_{n \rightarrow \infty} W^n(B)$, $\forall B \in \mathcal{H}(X)$. 此式表明, 吸引子 A 可以由变换 W 迭代作用于任意初始集 B 来生成, 且结果与初始集无关. 因为 $(\mathcal{H}(X), d_H)$ 是完备空间, 所以 $A \in \mathcal{H}(X)$.

式 (7.3.2) 表明, 吸引子 A 满足自拼贴性质 $A = \bigcup_{i=1}^N w_i(A)$. 也就是说, 吸引子 A 由它在变换 w_i 下的象 $w_i(A)$ “拼贴” 而成, 或者说吸引子 A (整体) 与 $w_i(A)$ (部分) 具有自相似性. 因此, 这个吸引子具有分形结构, 或者说 A 是一个分形.

所谓迭代函数系统 (IFS) 的正问题, 就是已知 IFS 如何构造 IFS 吸引子的问题. 下面, 我们讨论如何利用计算机绘制直观的 IFS 吸引子. 可供选择的算法很多^[16~18], 确定算法、随机算法是最有代表性的两种方法. 有兴趣的读者可以参考 Hepting 的综述文章^[19], 文章综述了各种算法, 并提出了一些有待解决的问题.

(一) 确定算法

这是直接按式 (7.3.1) 来计算的. 首先在 \mathbb{R}^2 平面上任意选择一个集合 A_0 , 可以是矩形、单位圆等等. 然后顺序地使用每个压缩仿射变换 w_i 得到 $w_i(A_0)$, $i = 1, 2, \dots, N$, 并令 $A_1 = \bigcup_{i=1}^N w_i(A_0)$, 再对 A_1 顺序使用每个压缩仿射变换得到 $A_2 = \bigcup_{i=1}^N w_i(A_1)$, 如此反复, 将得到一个集合序列 $\{A_n\}_{n=1}^{\infty}$, $A_n = \bigcup_{i=1}^N w_i(A_{n-1}) \in \mathcal{H}(X)$. 前面的讨论告诉我们, 该集合序列一定收敛于 IFS 的吸引子.

下面例子给出的示例形象地显示这个算法执行的过程 (参看图 7-2 和图 7-3).

例 7.3.1 Sierpinski 三角的手工绘制 (图 7-2).

考虑迭代函数系统:

$$\{R^2; w_1, w_2, w_3\}, \quad W(A) = \bigcup_{i=1}^3 w_i(A), \quad (7.3.3)$$

其中, 三个压缩仿射映射作用由图 7-1 给出. 图 7-2 显示了由式 (7.3.3) 定义的分形变换 W 迭代作用于一个初始集 (黑白蝴蝶图) 产生的集合序列收敛于 Sierpinski 三角的过程.

例 7.3.2 另一个 Sierpinski 三角的手工绘制 (图 7-3).

考虑另一个迭代函数系统:

$$\{R^2; w_1, w_2, w_3\}, \quad W(A) = \bigcup_{i=1}^3 w_i(A), \quad (7.3.4)$$

其中, 三个压缩仿射映射作用由图 7-3 给出. 图 7-4 显示了式 (7.3.4) 定义的 IFS 变换 W 迭代作用于一个初始图像 (全黑图像) 产生的集合序列收敛于另一种 Sierpinski 三角的过程.

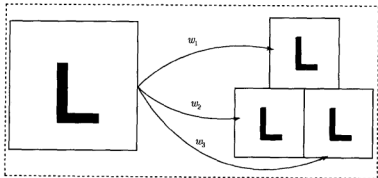


图 7-1 三个压缩仿射映射的作用

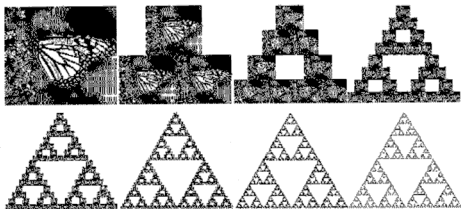
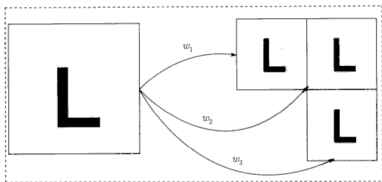
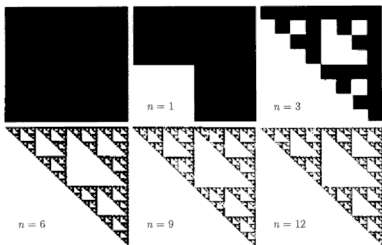

 图 7-2 集合序列 $W^n(B)$, 其中 B 是黑白蝴蝶图


图 7-3 三个压缩仿射映射的作用


 图 7-4 集合序列 $W^n(B)$, 其中 B 是全黑图

上述朴素的手工算法 (亦见第六章的多功能缩印机) 仅有理论意义, 实用性不大。用确定算法绘制 IFS 吸引子图形必须采用下面的方法, 其具体步骤如下:

Step 1 初始化: 任意给定一些点, 并设定最大迭代步数;

Step 2 对这些点顺序计算它们经过各个压缩仿射变换 $w_i (i = 1, 2, \dots, N)$ 变换后的象点并保存;

Step 3 擦去屏幕上原来的点, 打上 (2) 中计算出的点;

Step 4 返回到第二步, 直到迭代达到最大步数为止。

下面是上述算法实现的两个实例 (目前 Internet 网上已有许多现成的软件可以绘制图 7-5 和 7-6)。

例 7.3.3 用 IFS 逼真地再现枫叶、龙和树等自然图形。迭代生成 7-5 和图 7-6 的仿射变换分别在表 7-2 和表 7-3 中给出。



图 7-5 枫叶



图 7-6 分形树

表 7-2 生成枫叶 (图 7-5) 的四个仿射变换

	a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2
枫	0.6	0	0	0.6	0.18	0.36
叶	0.6	0	0	0.6	0.18	0.12
(Maple leaf)	0.4	0.3	-0.3	0.4	0.27	0.36
	0.4	-0.3	0.3	0.4	0.27	0.09

表 7-3 生成分形树 (图 7-6) 的四个仿射变换^[19]

	a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2
分	0.53	-0.08	0.08	0.53	-0.88	33.44
形	-0.31	-0.42	-0.44	0.33	-15.19	19.43
树	-0.25	-0.05	-0.07	0.29	1.48	11.73
(Fractal tree)	0.29	0.54	-0.04	0.29	18.74	9.87

(二) 随机算法

这是一种随机地选取 IFS 中迭代映射的方法. Barnsley 把这种算法称为混沌游戏算法 (chaos game algorithm), 它是目前计算机绘制 IFS 吸引子的一种最有效的算法. 与通常的迭代算法不同, 混沌游戏算法的目标不是迭代轨迹的极限, 而是迭代轨迹本身. 我们不去讨论这个算法的理论依据, 如收敛性、遍历性等, 它们保证了算法的有效性, 即只要迭代次数足够多, 算法几乎肯定能画出 IFS 吸引子. 这里只提出算法的具体操作.

首先, 对 X 上的每个仿射映射 w_i 附上一个概率 $p_i (i = 1, 2, \dots, N)$, IFS 变成含概率的 IFS (称为随机 IFS), 即

$$\{X; w_i, p_i, i = 1, 2, \dots, N\}, \quad (7.3.5)$$

其中, 概率的选择必须满足:

$$p_i > 0, \quad \sum_{i=1}^N p_i = 1, \quad (7.3.6)$$

一般可以按下面的方法给定:

$$p_i = \frac{|\det(A_i)|}{\sum_{i=1}^N |\det(A_i)|} = \frac{|a_{11}a_{22} - a_{12}a_{21}|}{\sum_{i=1}^N |a_{11}a_{22} - a_{12}a_{21}|}. \quad (7.3.7)$$

因为对某些 i , 有可能 $\det(A_i) = 0$ (即 w_i 不可逆), 实际操作时可把这些 p_i 取为较小的数, 比如 0.001.

还有一点与确定算法不同: 起始图像不再是一组点, 而是任意一点 $x_0 \in X$. 然后按概率 p_i 选择下一个点 x_1 , 余下类推.

随机算法的具体步骤如下:

- (1) 任意给定一个初始点 x_0 , 并设定总迭代步数;
- (2) 按概率 p_i 选择仿射映射 $w_i (i = 1, 2, \dots, N)$, 得到下一个点 $x_1 = w_i(x_0)$;
- (3) 令 $x_0 = x_1$, 如果迭代次数大于 10, 则在屏幕上打出点 x_0 ;
- (4) 重返第 (2) 步, 进行下一次迭代, 直到迭代步数大于总步数为止.

在第 (4) 步中, 设定第 10 次迭代后再打点, 是为了消除起始点可能不在吸引子上的初始迭代点, 这个值是预设的, 也可以再取大些. 随着迭代步数的增加, 轨迹点将收敛到吸引子上来. 以此方法生成表 7-4 表示的 IFS 的吸引子如图 7-7 所示.

表 7-4 生成图 7-7 的四个仿射变换及概率

	a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2	p
蕨	0	0.3535	0	0.16	0	0	0.01
叶	0.85	0.04	-0.04	0.85	0	1.6	0.85
(Fern leaf)	0.2	-0.26	0.23	0.22	0	1.6	0.07
	-0.15	0.28	0.26	0.24	0	0.44	0.07



图 7-7 蕨叶

第四节 迭代函数系统逆问题与图像编码

我们已经看到, IFS 建模具有典型分形特征的自然图形确实是十分成功的, 其蕴涵的视觉信息完全“封装”到简单的 IFS 码中. 不少看似复杂的分形图形, 从计算的观点看来, 其信息量并不大, 可以通过迭代这一全反馈的动态过程用参数不多的压缩仿射变换来产生. 显然, 存储这些复杂图形的分形码 (变换参数) 当然要比存储图形本身节约空间得多. 以图 7-7 为例, 图中的树看起来复杂得难以用传统方法描述, 但它却生成于四个简单的仿射变换. 若按像素存储这棵树, 以分辨率 550×480 计, 需要 264,000 比特. 即使不考虑有效存储方法, 代之与存储相应的 IFS 码, 压缩比肯定在 1000:1 以上^①.

于是, 从相反的方向考虑上述问题便是自然而然的想法, 即用参数不多的压缩变换表达现实图像, 并用简单的迭代过程重构原图像. 这就产生了图像的分形编码问题. 具体说, 对于给定图像, 寻找一个 IFS 使得吸引子就是原图像或原图像的近似. 这就是分形编码文献中常常提到的 IFS 逆问题, 也就是自然图像的分形编码. 如果找到了这样的 IFS, 只需存储这些图像的 IFS 码 (IFS 的参数) 而不是存储图像本身, 则能够实现图像的超高倍压缩. 这个新颖想法为图像压缩提供了一条与以往完全不同的新思路, 从而使图像编码出现一个崭新的领域.

^① 应该指出, 上述 IFS 码的输出分辨率是可以选择的, 如 225×240 或 1100×960 . 按照上面的计算方法, 得到“无穷大”压缩比是不足为奇的, Barnsley 所称的 10000:1 的超高压缩比似乎还太保守. 但是, “压缩比”应该从另一个方向来计算, 即给定一幅数字图像, 寻找一个 IFS 码, 其解码图像 (吸引子) 接近待编码图像, 然后用这个 IFS 码计算压缩比. 这样得到的压缩比才能算作“真正”的压缩比. 但事实证明, 早期的分形编码方法在寻找吸引子接近现实图像 (如人脸图像) 的 IFS 码方面是不成功的. 因此, 我们应该正确看待分形编码文献常常提及的 10000:1 的超高压缩比.

但是要实现上面的想法, 有两个问题不可回避. 这两个问题是:

(1) 分形变换 W 的不动点 (或 IFS 的吸引子) 有什么性质? 现实图像具有这样的性质吗?

(2) 对于给定的图像, 如何寻找它的 IFS 码? 有计算机自动进行的算法吗?

对于第一个问题, 让我们再回过头去看图 7-2, 不难发现, 变换 W 对初始图像的每一次作用都产生更精细尺度下的细节, 如果作用无限进行下去, 极限图像 (变换 W 的不动点) 将具有无限精细的结构. 此外, 这个极限图像还具有不同尺度下的仿射自相似性. 换句话说, 这个极限图像是分形. 因此, 实现上述想法的第一步就是假设现实图像是分形, 或者说把待编码图像近似视为一个分形图像. 大自然“处处分形” (分形理论的基本观点) 是其立论的基础.

第二个问题是分形编码技术最基本的问题, 在分形编码中称为 IFS 逆问题 (即图像的 IFS 编码). 我们先给出它的一般性描述: 设 (X, d) 是一个数字图像空间 (完备度量空间), 度量 d 描述两个图像之间的差异 (失真测度). $F(X)$ 是 IFS 的集合, $\mu_{\text{org}} \in X$ 是待编码图像, 求一个 IFS 码 W_{opt} , 使得

$$W_{\text{opt}} = \arg \min_{W \in F(X)} d(\mu_{\text{org}}, \mu_A), \quad (7.4.1)$$

其中 μ_A 是 IFS 码 W_{opt} 的吸引子. 式 (7.4.1) 意指 W_{opt} 是吸引子 μ_A 与待编码图像 μ_{org} 差异最小的 IFS 码. 因此, “寻找图像的 IFS 码” 就是求解优化问题 (7.4.1).

我们知道, 吸引子可以从 Banach 不动点定理给出的迭代算法快速迭代生成, 但是它通常不能由 IFS 参数简单地表示出来. 因此, 与数学中许许多多的逆问题一样, 准确求解 IFS 逆问题 (7.4.1) 已证明是十分困难的. 尽管已有许多学者用各种方法研究 IFS 逆问题的求解, 如 Fourier 变换^[20]、小波变换^[21]、矩方法 (moment method)^[22~25]、遗传算法^[26]、神经网络^[27]、小波变换与矩方法的结合^[28]、模糊集^[29]、泛函方法^[30] 以及其他方法^[31,32], IFS 逆问题仍然没有得到圆满解决. 此外, 已经证明确定最优分形码的问题是 NP- 难的^[33].

那么, 如何寻求一个不动点图像是待编码图像较好逼近的 IFS 呢? 这是分形图像编码技术必须首先解决的最基本的问题. 目前, 借助拼贴定理是部分解决这个问题折中办法, 因为按此方法只能得到次优解. 我们已在第六章中就一般的度量空间介绍过拼贴定理. 在分形空间 $\mathcal{H}(X)$ 中, 拼贴定理表现为下面的形式:

$$d_H(C, A) \leq \frac{1}{1-s} d_H(C, W(C)), \quad \forall C \in \mathcal{H}(X), \quad (7.4.2)$$

其中 W 是压缩因子为 s 的压缩变换, A 是其吸引子.

不等式 (7.4.2) 告诉我们这样的事实: 要使 A 与 C 接近, 只要 $W(C)$ 与 C 接近, 即 $W(C) = \bigcup_{i=1}^N w_i(C) \approx C$. 因此, 在分形编码文献中, 常常把 $W(C)$ 形象地称为

C 的“拼贴”，即 C 是由诸 $w_i(C)$ 拼贴而成。于是，我们可以按极小化 $d_H(C, W(C))$ 的方式寻找变换 W 。这就是 Barnsley 提出的 IFS 编码方案的基本思想。

IFS 编码方案的解码原理由式 (7.4.3) 给出。变换 W 迭代作用于任意的初始图像 B ，直到 $W^n(B)$ 没有明显变化为止（收敛）：

$$\left. \begin{aligned} C &\approx W(C) \\ A &= \lim_{n \rightarrow \infty} W^n(B), \forall B \end{aligned} \right\} \Rightarrow \left. \begin{aligned} C &\approx A && \text{by (3.8)} \\ A &\approx W^n(B), n \text{ large enough} \end{aligned} \right\} \Rightarrow \left. \begin{aligned} C &\approx W^n(B), \forall B, \\ &n \text{ large enough} \end{aligned} \right\} \quad (7.4.3)$$

待编码图像 C 的拼贴 $C \approx \bigcup_{i=1}^N w_i(C)$ 可以按下面的方式进行：先把图像 C 分割为 N 个不重叠的部分 $C = \bigcup_{i=1}^N C_i$ ，要求每一块 C_i 与整幅图像的变换象 $w_i(C)$ 尽量相似，即 $C_i \approx w_i(C)$ （这需要靠人眼来识别）。例如，假定我们要编码图 7-2 显示的 Sierpinski 三角，对 Sierpinski 三角的观察，使我们想到图 7-1 给出的变换。但对现实图像，靠人眼来识别这样的相似显然是难以办到的。

因此，不难看出上述 IFS 编码方案的局限：

(1) 图像的整体与局部要有很强的仿射自相似性 易于 IFS 生成的图像都是一些特殊的类型（如 Sierpinski 三角），图像的局部与整体要有很强的仿射自相似性，它们看起来很像图像本身经旋转、反射、伸缩、偏斜等操作后的扭曲“复制品”的拼贴。现实图像很少具有这样的性质，即使有，也很难靠人眼来识别。

(2) 人机交互式编码 对于有很强的仿射自相似性的图像，这些自相似性是靠人眼来识别的，图像编码是靠人机交互方式设计完成的，或者说编码过程中需有人工干预，即原图像的拼贴如何进行，需要人的参与，这在技术上不可行。

尽管图像的 IFS 编码方案有上述局限，其原始形式在表达现实图像方面是不足的，但 IFS 还是为生成计算机图形提供了一个强有力的工具。庆幸的是，修改后的 IFS 较好地克服了上述不足，从而使分形压缩成为一门实用的技术。下一节将介绍这方面的内容。

第五节 分形编码算法的基本原理与实现

超越 Barnsley 编码方案的局限，Jacquin 提出了基于分块迭代函数系统 (Partitioned IFS) 的分形块编码算法。这是首次利用计算机进行数字图像的分形压缩的自动算法，对分形图像压缩方法的实用化起了奠基的作用。

分形图像编码 PIFS 方法，本质上是假设自然图像中不同区域间存在着跨尺度冗余 (across-scale redundancy)——不同尺度下的冗余来实现图像压缩的。对于自然图像，我们不难找出其不同区域间确实存在这种不同尺度下的相似性，例如照片

中不同远近、不同位置的物体,如标准测试图像“boat”中的桅杆(图 7-8).此外,仔细观察标准测试图像“Lena”也不难发现这个事实:肩膀处嵌套的两个区域、帽沿处的小区域与右下角的大区域,两个大区域缩小一定比例就分别与各自的小区域基本吻合(图 7-9).

从结构上看,PIFS 码与 IFS 码几乎是相同的.唯一的差别是,前者是利用图像的不同区域在不同尺度下的相似性(局部与局部的相似性),后者是利用图像的不同区域在不同尺度下与图像本身的相似性(局部与整体的相似性).



图 7-8 Boat 图像中的自相似区域



图 7-9 Lena 图像中的自相似区域

一、基本原理

设 $(\mathbb{R}^{N \times N}, d)$ 是 $N \times N$ 灰度图像空间,灰度值范围是 $\{0, 1, 2, \dots, l-1\}$ (l 一般为 256, 即 8 bit 量化). 因此,一幅图像 I 可以表达为一个矩阵 $(I_{ij})_{N \times N}$, I_{ij} 表示图像在 (i, j) 处的灰度值. d 是用作失真判据 (distortion criteria) 的完备度量,在分形编码文献中常常选择为均方根 (RMS, Root Mean Square):

$$d(x, y) = \text{RMS}(x, y) = \left(\frac{1}{N^2} \sum_{i,j=1}^N |x_{ij} - y_{ij}|^2 \right)^{1/2}, \quad x, y \in \mathbb{R}^{N \times N} \quad (7.5.1)$$

上式定义的度量与数学文献中的欧式度量仅仅差常数因子 $1/N$.

为了更好地理解 PIFS 编码,把灰度图像 (image) 看成二元函数 $z: S \rightarrow \mathbb{R}$, $(x, y) \mapsto z(x, y)$ 的图像 (graph) 是方便的. 其中 $S \subset \mathbb{R}^2$ 是灰度图像的支持或背景, $z(x, y)$ 是像素点 (x, y) 的亮度或灰度值,它取有限个非负值 (一般为 $0 \sim 255$). 例如,在图 7-10 中,左图是标准测试图像 256×256 Lena 图像,右图是对应的二元函数图像.

此外,为了实现图像压缩,变换 W 至少应该满足两个条件:

- (1) 它的不动点(吸引子)能够较好地逼近现实图像;
- (2) 能够紧凑地编码(code compactly).

显然, 仿射变换能够很好地满足这些要求.

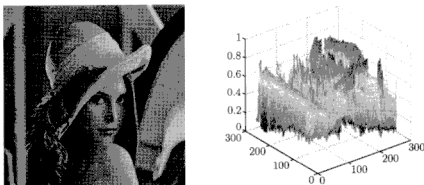


图 7-10 Lena 图像被视为二元函数的图像

分形编码的任务就是要寻找一个压缩变换 W , 使得 W 的不动点尽可能接近待编码图像. 在大多数分形编码器中, 在最简情形(固定方块分割), 图像被分割成大小两类子块(例如大块 D_i 为 8×8 , 小块 R_i 为 4×4), 小块 R_i 互不重叠且覆盖整幅图像(图 7-11). 在分形编码文献中, 小块 R_i 称为 range 块(以下简称 R 块), 大块 D_i 称为 domain 块(以下简称 D 块). 具体说, 把待编码图像 $I(N \times N$ 像素)划分为 $N_r \cdot N_r$ 个互不重叠的 $B \times B$ 像素块 R_i (图 7-11):

$$I = \bigcup_{i=1}^{N_r \cdot N_r} R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j, \quad (7.5.2)$$

其中, $N_r = N/B$. R_i 的集合构成 R 块池. D 块 D_i 的集合构成 D 块池, D_i 的尺寸一般选取为 $2B \times 2B$ 像素, 可以有重叠, 相邻域块 D_i 的移位为 δ 个像素点(δ 也称为步长). 因此, D 块的个数为 $N_d = \left(\frac{N-2B}{\delta} + 1\right)^2$. δ 一般取为 B (或 $2B$), 此时 D 块的个数为 $(N_r - 1)^2$, 且在纵(横)方向上是半重叠的. 每个 R 块 R_i 由某个 D 块 $D_{m(i)}$ 的大小比例重定与亮度变换来近似(图 7-12). 具体说, 对于 $B \times B$ 像素 R 块 R_i , 通过某种方法找到与其最佳匹配的 $2B \times 2B$ 像素 D 块 $D_{m(i)}$. 块变换 $w_i: D_{m(i)} \rightarrow R_i$ 是 D 块到 R 块的变换, 它由两个变换复合而成: $w_i = \lambda_i \circ \gamma_i$ (图 7-12 给出了示意图). 它们的全体构成了块变换库 $\mathcal{W} = \{w_i, i = 1, 2, \dots, N_r\}$. 这些变换以分片方式定义了一个作用于整幅图像的变换(称为分形变换):

$$\forall I \in \mathbb{R}^{N \times N}, \quad W(I) = \sum_{0 \leq i \leq N_r^2} (W(I)|_{R_i}) = \sum_{0 \leq i \leq N_r^2} w_i(I|_{D_{m(i)}}), \quad (7.5.3)$$

这里, $g|R$ 表示图像 g 在 R 上的限制. 式 (7.5.3) 的涵义是: 图像 I 的每个 R 块 $I|R_i$, 用图像 I 的与之匹配的 D 块 $D_{m(i)}$ 在块变换 w_i 下的像 $w_i(I|D_{m(i)})$ 代替, 这样拼贴出的新图像就是 $W(I)$.

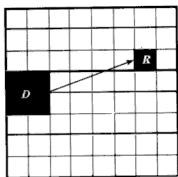
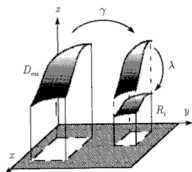


图 7-11 PIFS 编码的图像分割

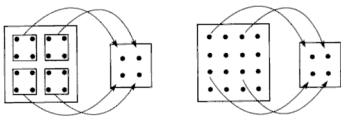

 图 7-12 变换 w_i 的作用

从图 7-12 中可以看出, 映射 γ_i 把子图像 $D_{m(i)}$ (D 块) 平移到子图像 R_i (R 块) 的位置, 然后收缩使它的支持与 R_i 的支持完全重合, 并进行像素值平均或欠采样 (decimation). 像素值平均或欠采样的含义由图 7-13 给出. 映射 λ_i 修改灰度信息以得到 R_i 的较好近似, 允许灰度级调整、灰度平移 (图 7-14) 和像素的 8 种重排 (4 个旋转和 4 个翻转, 称为图像块的等距变换, 如图 7-15 所示). 其一般形式为

$$\lambda_i \begin{pmatrix} x \\ y \\ z(x, y) \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z(x, y) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ o_i \end{pmatrix}, \quad (7.5.4)$$

其中, $-a, b, c, d$ 决定了像素的 8 种重排 (4 个旋转和 4 个翻转);

$-s_i, o_i$ 分别代表亮度调整 (scaling) 和亮度偏移 (offset) (图 7-14)


 图 7-13 收缩 D 块的构造, 像素值平均 (左) 和欠采样 (右)

必须指出, 正是参数 s_i, o_i 的引进, 才使仿射变换后的 D 块的灰度有可能匹配 R 块的灰度 (图 7-14). 对此我们做个类比以理解这两个参数的作用: 参数 s_i 相

当于电脑显示器的对比度调整按钮, s_i 为 0 时, 屏幕被调整为全黑; s_i 为 1 时, 对比度未作调整; $0 < s_i < 1$ 时, 对比度降低; $s_i > 1$ 时, 对比度增强. 参数 o_i 相当于显示器的亮度调整按钮, 正的 o_i 使屏幕变亮; 负的 o_i 使屏幕变暗.

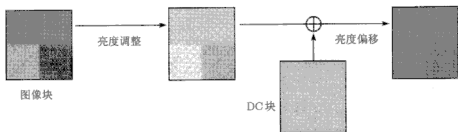


图 7-14 灰度级调整 (对比度调整) 与偏移 (亮度调整)

映射 w_i 的一般形式为

$$w_i(D_{m(i)}) = \lambda_i(\gamma_i(D_{m(i)})) = s_i \cdot t_k(\gamma_i(D_{m(i)})) + o_i, \quad (7.5.5)$$

其中, t_k 是等距变换 ($k = 0, 2, \dots, 7$), 分别是文献 [4] 引入的恒等变换和 90° 、 180° 和 270° 逆时针旋转以及关于 x 轴方向、 y 轴方向、 45° 线和 135° 线对称反射 (图 7-15). 图 7-16 显示, 等距变换的作用是十分明显的. 图 7-16 显示了两层意思: (1) 容许像素重排. D 块 D_1 与 R 块 R 直接是不相似的, 但只要先作旋转运算 (逆时针旋转 90°), 再缩小一半, 结果与 R 完全吻合; (2) 不容许像素重排. 假设图像中与 R 块 R 最匹配的 D 块共有两个, 即图中 D_1 和 D_2 , 因为不容许像素重排, R 只好匹配 D_2 , 这显然不如前一种匹配.

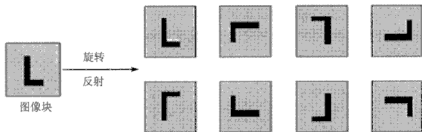
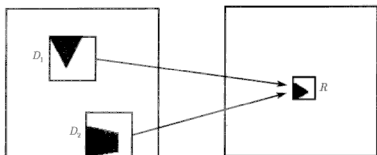
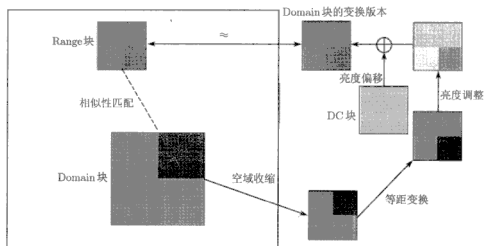


图 7-15 图像块的 8 种旋转和翻转

为了更好地理解块变换 w_i , 我们特设计了图 7-17, 它形象地显示了式 (7.5.5) 定义的块变换 w_i 的作用情况. 图中, D 块经空域压缩以匹配 R 块尺寸, 接着施行等距变换 (此处为顺时针旋转 90°)、亮度调整, 然后加一个 DC 块 (亮度偏移), 得到的图像块近似等于 R 块.


 图 7-16 D 块 D_1 在映射到 R 块 R 前, 应该先作旋转运算

 图 7-17 块变换 w_i 作用示意图

为了压缩图像, 我们必须回答下面的逆问题: 如何划分图像以及如何选择块变换 w_i 的参数, 使分形变换 W 的不动点接近待编码图像? 现有的编码方案是基于拼贴定理:

$$d(I, F) \leq \frac{1}{1-s} d(I, W(I)), \quad \forall I \in \mathbb{R}^{N \times N}, \quad (7.5.6)$$

其中, I 是待编码图像, F 是分形变换 W 的不动点, s 是变换 W 的压缩因子, d 是用作失真判据的完备度量。于是, 上述逆问题转换为“寻找给定点与其映像接近的压缩变换”的问题, 即极小化拼贴误差 $d(I, W(I))$ 。此外, 为保证收敛, 一般要求分形变换是压缩的。于是拼贴定理保证其不动点图像 (吸引子) 逼近待编码图像。

具体的说, 给定待编码数字图像 I 、R 块池、D 块池和块变换库 \mathcal{W} , 对于每个 R 块 R_i , 按照某种搜索方法寻找 D 块 $D_{m(i)}$ 和块变换 $w_i \in \mathcal{W}$, 使得 $R'_i = w_i(D_{m(i)})$

与 R_i 在取定的失真度量 d 下尽可能接近, 即

$$d(R_i, w_i(D_{m(i)})) = \min\{d(R_i, w(D)) | w \in W, \forall D\}, \quad (7.5.7)$$

其中, d 取 (7.5.1), 也可以取其他的失真度量.

问题 (7.5.7) 是一个复杂的非线性优化问题, 计算复杂度非常高. 目前, 绝大多数分形图像编码论文的主题都集中在这个方面.

当最优块变换 w_i 及 D 块 $D_{m(i)}$ 找到之后, 存储或传输其量化参数, 完成 R 块 R_i 的编码. 待所有 R 块被编码以后, 原图像的分形编码也就完成了.

在上述编码步骤中, 图像的分割方式 (决定 R 块 R_i 的形状)、 D 块的选取以及块变换库的建立方式对图像压缩的精度、计算复杂性和压缩效率等都有较大影响. 我们将在下一章中讨论这些问题.

总之, 在分形编码中, 一幅图像是用一个使图像近似不变的压缩变换的参数来表达的, 而这样的参数往往可以紧凑地编码. 这大大减少了储存图像的比特数, 因而实现了图像数据的高压缩比. 分形解码较为简单, 分形解码器是利用 Banach 不动点定理给出的迭代算法设计的, 即对于任意初始图像 μ_0 , 变换不动点 $f_W = \lim_{n \rightarrow \infty} W^n(\mu_0)$, W 是表达图像的分形变换. 于是, 待编码图像可按迭代方式简单地重构 (有一定的失真), 即待编码图像近似等于 $W^N(\mu_0)$, 其中 N 是迭代次数. 应该指出, 从严格的数学角度上讲, 这种迭代应该是无穷次的, 但在实际的数字图像的解码中, 由于分辨率的限制, 这种迭代只需有限次即可. 此外, 实验结果表明, 一般 $N \leq 10$ 可以满足要求.

最后指出, 基于 PIFS 的分形编码与基于 IFS 的分形编码在基本原理上是相同的, 即表达图像的变换是压缩的, Banach 不动点定理保证变换的不动点图像是存在唯一的, 且可以从任何初始图像迭代生成. 此外, 因变换使图像近似不变, 拼贴定理保证这个不动点图像是待编码图像的一个逼近. 但是, 二者在具体寻求变换的方式上是不同的.

二、算法设计基本步骤

我们在上节介绍了分形编码的基本原理. 不难看出, 这个基本原理仅仅为分形编码算法的具体设计与实现提供了一个基本框架, 还留有很大的活动余地. 事实上, 设计分形编码算法时, 我们必须考虑许多问题.

分形编码算法的设计要遵循一定的基本步骤, 这些步骤通常包含:

(1) 确定图像分割方案: 分形编码的基本思想就是寻找图像中不同区域间的跨尺度相似性, 因此, 一个好的分割方案应该反映出图像的这种相似性. 分形码是由分割信息与变换参数组成的, 不同分割方案占有的分割信息量是不同的. 从压缩观点看, 分割方案占有的信息量越少越好 (最好是零, 如固定方块分割), 但是, 如果

一个不规则分割带来了好的编码质量(失真测度意义下),花费一点编码成本是值得的。尽管使用固定块分割(如方块分割、三角分割、矩形分割),分形码仅由变换参数组成,无需存储或传输分割信息,但是,其缺点是显而易见的,因为它们不能自适应于图像细节(如飞机与蓝天的细节显然不同)。选择何种分割方案有时将对算法的性能、效率等产生很大的影响。

(2) D 块池的构成方案。谈到分形编码的缺点时,不外乎“编码耗时”。而编码阶段的时间主要花费在对每个 R 块搜索匹配的 D 块上。理论上可以作为候选的 D 块的数目一般是很大的。例如, $N \times N$ 图像中,任意尺寸方块数目的阶是 $O(N^3)$ 。于是,我们必须对可容许 D 块作某些限制,例如拿固定方块分割来说,我们必须限制 D 块的尺寸与位置等。但是,因为必须兼顾编码时间和编码质量,取一必舍二,如何找到对任意编码图像都适用的折中方案确实是一个艰难的决策。总之, D 块的构成方案对编码时间和编码质量都有很大影响,编码时务必谨慎决策。

(3) 变换类型的选择。在分形图像编码中,一幅图像 I 的编码就是寻找一个不动点与待编码图像最接近的压缩变换 T 的过程,而拼贴定理把这个过程转化为极小化 I 与 $T(I)$ 的距离。因此,尽管从原理上说,作用于 D 块的变换只要求是压缩的而无其他限制。但是出于计算上的考虑,我们必须对变换的类型添加一定的限制。一般说来,除压缩以外,变换还应该受到三个约束: 1) 变换不能是线性的,否则其不动点只能是零灰度图像; 2) 变换应该结构简单、计算方便,以简化拼贴误差优化、快速解码和简便分析; 3) 变换的不动点相对于参数量化应该具有鲁棒性,因为参数最后必须量化。仿射变换能够很好地满足这些要求,但是没有任何理由说仿射变换就是最好的选择。

(4) D 块搜索方案(与 D 块的构成方案密切相关)。以对 $N \times N$ 图像作固定方块分割为例,设 R 块尺寸 $B \times B$ 、D 块尺寸 $2B \times 2B$,相邻域块 D_i 的移位为 δ (决定了 D 块的位置),不难计算出 D 块的个数为 $N_d = [(N - 2B)/\delta + 1]^2$ 。因此,全搜索固然能够得到最好的编码质量,但全搜索的运算量大是不争的事实。如何减少计算复杂性,目前有许多解决方案,但归根结蒂,其原则无非是化全搜索为局部搜索。正如查字典一样,如果在整本字典中搜索某个字,工作量当然大。但是,如果事先决定字的部首,然后再在该部首内搜索,工作量就小多了。进一步还可以利用笔画的多少来减小搜索的工作量。

(5) 参数的表示、量化策略与比特分配。例如,为了得到最好的编码性能,应该采用多少比特量化灰度映射中的尺度系数和偏移系数。为了进一步压缩数据,可能还需要结合熵编码。

(6) 编程上机运行。

应该注意,上述各基本步骤是密切相关的,比较某一步骤里不同决策的编码性能,常常需要考虑其他步骤的相应决策。例如, D 块搜索方案就与 D 块池的构成紧密相关,编码时二者必须兼顾。遗憾的是,目前几乎没有理论结果能指导我们做

出相应决策,只能凭经验或仿真试验.下节中,我们从 VQ 观点介绍具体的算法及实现.

三、基本算法描述与实现

基本分形编码算法曾在绪论中简单介绍过,它基于对图像进行最简单的固定方块分割,本节将详细讨论这个算法.需要指出,这种算法是分形图像编码中最基本、最简单的形式,设计分形压缩软件时是不会使用这种简单形式的.但它有简单易懂、又能显示分形编码实质的优点,是介绍分形编码算法的最好教材,在许多分形编码文献中也常常用来显示新算法的改进之处.

基本分形编码与 VQ 编码十分类似.在编码阶段,两种方法都要搜索码本以近似原始图像分割后的每个子块.二者不同之处主要在于生成码本的方式不同.在 VQ 编码中,码本生成于一组训练图像,并用于编码未必是训练图像的任何图像,但为了重构原始图像,解码器也必须有一本同样的码本,或者说码本是离线传输的.在分形图像编码中,码本源于原始图像,而且解码器并不存在也不需要这本码本(或者说码本是无需传输的),因为分形解码是迭代解码,解码器只需按照分形文件描述的压缩变换迭代作用于任何同尺寸图像即可恢复原始图像.这种类似性使得较为成熟的 VQ 技术中许多编码方案可以稍加修改就可以移植到分形编码中,正如许多分形编码文献所作的那样.鉴于上述原因,我们认为,从 VQ 观点介绍分形编码算法及实现是一个好的选择.

基本分形算法的编码过程由下列几部分组成:

(一) 图像分割

假设待编码图像 μ_{org} 是 $N \times N$ 灰度图像,每像素灰度 8 比特量化(即把灰度分为 256 级).应用中, N 一般为 2 的方幂,例如 256×256 、 512×512 等.采用固定方块分割方法,把图像 μ_{org} 分割成一系列尺寸固定的 $B \times B$ 像素子块(二维阵列) $R_i (i = 1, 2, \dots, N_r)$,它们互不重叠且覆盖整幅图像(图 7-18),也就是说,

$$\mu_{\text{org}} = \bigcup_{i=1}^{N_r} R_i, \quad R_i \cap R_j = \emptyset \quad (i \neq j), \quad i, j = 1, 2, \dots, N_r. \quad (7.5.8)$$

在分形编码文献中,这种子块称为 range 块(以下简称 R 块),应用中其大小一般为 4×4 、 8×8 和 16×16 等.在 R 块组成的子块中,通常按行序逐块编码,即把 R 块排序为 $R_{11}, R_{12}, \dots, R_{1n}, R_{21}, R_{22}, \dots, R_{n1}, R_{n2}, \dots, R_{nn} (n = N/B)$.

此外,图像 μ_{org} 又划分成一系列尺寸大一些的子块 $\{D_i\}_{i=1}^{N_d}$,它们可以有重叠且不必覆盖整幅图像,并称它们为 domain 块(以下简称 D 块).应用中,对应于尺寸为 $B \times B$ 的 R 块, D 块尺寸一般为 $2B \times 2B$ (如 8×8 、 16×16 和 32×32 像素

等). 它们可以在图像中以水平步长 δ_h 和铅垂步长 δ_v 从左到右、自上而下滑动一个 $2B \times 2B$ 窗口来生成. 显然, 相邻两块之间在水平方向 (或铅垂方向) 有 δ_h (或 δ_v) 个像素的重叠. 应用中, 水平步长与铅垂步长一般为等长 $\delta_h = \delta_v = \delta$, 且为 R 块边长 B (如图 7-19 所示, 图中画出了三个 D 块).

R_{11}	R_{12}	...	R_{1n}
R_{21}	R_{22}	...	R_{2n}
\vdots	\vdots	...	\vdots
R_{n1}	R_{n2}	...	R_{nn}

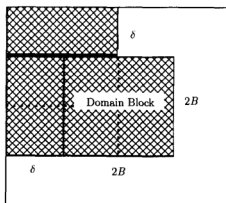
图 7-18 分割方案 (R 块)

图 7-19 生成 Domain 块

(二) 码本构成

对于每个 D 块 $D_j (j = 1, 2, \dots, N_d)$, 采用四邻域像素值平均或欠采样 (如图 7-13 所示), 得到 $B \times B$ 像素块 \hat{D}_j , 用 S 表示这种运算 (空域收缩算子), 即 $S(D_j) = \hat{D}_j$. 例如按像素表达, 四邻域像素值平均为

$$\hat{d}_{k,l} = (d_{2k,2l} + d_{2k+1,2l} + d_{2k,2l+1} + d_{2k+1,2l+1}) / 4, \quad (7.5.9)$$

其中, $d_{k,l}$ 、 $\hat{d}_{k,l}$ 分别是 D_j 和 \hat{D}_j 在像素点 (k,l) 的灰度值. 这种收缩子块的全体就构成“虚拟码本”, 记这个码本为 Ω , 即 $\Omega = \{\hat{D}_j = S(D_j) : j = 1, 2, \dots, N_d\}$.

以下为记号简单, 我们将仍用 D 表示码本 Ω 中的子块.

(三) 等距变换

为了改进图像质量, 空域收缩后的 D 块 \hat{D}_j 需要进行 8 个等距变换 t_k . 为叙述方便, 对于子块 $D \in \Omega$, 我们用 $D_{i,j}$ 表示它在位置 (i,j) 处的像素值, 于是, 8 个等距变换可按像素形式表达如下^[4]:

(1) 恒等变换: $(t_0 D)_{i,j} = D_{i,j}$;

(2) 关于铅垂中轴 ($j = (B-1)/2$) 的对称反射:

$$(t_1 D)_{i,j} = D_{i,B-1-j};$$

(3) 关于水平中轴 ($i = (B-1)/2$) 的对称反射:

$$(t_1 D)_{i,j} = D_{B-1-i,j};$$

(4) 关于主对角线 ($i = j$) 的对称反射:

$$(t_3 D)_{i,j} = D_{j,i};$$

(5) 关于次主对角线 ($i + j = B-1$) 的对称反射:

$$(t_4 D)_{i,j} = D_{B-1-j, B-1-i};$$

(6) 关于块中心逆时针旋转 90° :

$$(t_5 D)_{i,j} = D_{j, B-1-i};$$

(7) 关于块中心逆时针旋转 180° :

$$(t_6 D)_{i,j} = D_{B-1-i, B-1-j};$$

(8) 关于块中心逆时针旋转 270° :

$$(t_7 D)_{i,j} = D_{B-1-i,j}.$$

(四) 误差度量

为了定量化图像子块间的相似性, 必须选取合适的度量. 在分形编码文献中, 误差度量通常为均方误差 (Mean Square Error, MSE). 对于任意两个子块 $R, D \in \mathbb{R}^{B \times B}$, 其 MSE 定义为

$$\text{MSE}(R, D) = \|R - D\|^2 / B^2 = \sum_{i,j=1}^N |R_{i,j} - D_{i,j}|^2 / B^2 \quad (7.5.10)$$

其平方根是 $\mathbb{R}^{B \times B}$ 的一种完备度量, 称为均方根 (RMS, Root Mean Square), 它与数学文献中的欧式度量仅仅差常数因子 $1/B$. 此外, $N \times N$ 原始图像 μ_{org} 和重构图像 $\hat{\mu}_{\text{org}}$ 之间的 MSE 定义为

$$\text{MSE}(\mu_{\text{org}}, \hat{\mu}_{\text{org}}) = \sum_{i,j=1}^N |(\mu_{\text{org}})_{i,j} - (\hat{\mu}_{\text{org}})_{i,j}|^2 / N^2$$

由此得到测试解码图像 (8 bit 量化) 质量的一种重要参数 —— 峰值信噪比 (Peak Signal-to-Noise Ratio, PSNR):

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}(\mu_{\text{org}}, \hat{\mu}_{\text{org}})} \right) (\text{dB}). \quad (7.5.11)$$

(五) 编码过程

编码阶段, 对于每个 R 块 R_i , 我们要按照某种方法 (如全搜索) 在码本 Ω 中寻找一个最佳匹配块 $D_{m(i)}$, 以及对 $D_{m(i)}$ 的对比度和亮度的最优调整因子 s_i 和 o_i , 使得 R_i 与 $D_{m(i)}$ 的最优亮度变换版本 $\hat{D}_{m(i)} = s \cdot D_{m(i)} + o \cdot \mathbf{1}$ 之间的差异 $\text{MSE}(R_i, \hat{D}_{m(i)})$ 是最小的 (为了叙述方便, 我们暂不考虑等距变换), 即对于每个 R 块 R_i , 我们需要求解下面的约束极小化问题:

$$\min_j \left\{ \min_{s, o \in \mathbb{R}, |s| < 1} \|R_i - (s \cdot D_j + o \cdot \mathbf{1})\|^2 \right\}, \quad (7.5.12)$$

其中, $\mathbf{1}$ 是亮度值均为 1 的常值块, s 和 o 分别起调整 D_j 的对比度和亮度的作用。此外, 要求参数 s 满足约束 $|s| < 1$ 是为了理论上保证解码迭代序列收敛。

设约束极小化问题 (7.5.12) 的解为 $\{m(i), s_i, o_i\}$, 也就是说, $m(i)$ 是 R_i 的最佳匹配块 $D_{m(i)} \in \Omega$ 的序号, s_i 和 o_i 分别是对最佳匹配块 $D_{m(i)}$ 的对比度和亮度的最优调整, 且

$$\text{MSE}(R_i, \hat{D}_{m(i)}) = \frac{1}{B^2} \min_j \left\{ \min_{s, o \in \mathbb{R}, |s| < 1} \|R_i - (s \cdot D_j + o \cdot \mathbf{1})\|^2 \right\}. \quad (7.5.13)$$

于是, 每个 R 块 R_i 由其最佳匹配块 $D_{m(i)} \in \Omega$ 的最佳亮度变换 (对比度和亮度调整) 来近似: $R_i \approx s_i \cdot D_{m(i)} + o_i \cdot \mathbf{1}$ 。三元组 $\{m(i), s_i, o_i\}$ 称为 R_i 的分形码, 全体 R_i 的分形码就组成原始图像 μ_{org} 的分形码, 它描述了一个使 μ_{org} 近似不变的压缩仿射变换 T 。如果考虑等距变换^①, 则 R_i 的分形码则为四元组 $\{k, m(i), s_i, o_i\}$, 其中 k 是等距变换的序号。

问题 (7.5.13) 的内层极小化问题是一个约束极小化问题, 直接求解十分耗时, 加上其外层的极小化问题更加耗时 (需要匹配搜索, 占据了主要的编码时间), 完全可能使分形算法因编码时间太长而失去实用性。为了解决这个问题, 一般采用下面的两种次优的方法之一。

一种方法是借鉴乘积 VQ 的思想 (见 7.2.2 节), 视码本 Ω 为形态码本, 并事先构造对比度因子 s 的数量码本, 例如 $C_s = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, 亮度调整因子 o 则由 R 块的亮度均值编码。分形块编码算法的提出者 Jacquin 就是采用这种方法。但是这种方法仍然十分耗时, 为了较少编码时间, Jacquin 先把原始图像 μ_{org} 等分为四个部分, 并分别编码每个部分。但编码时间的减少是以牺牲图像质量为代价的。本文不采用这种方法。对此方法有兴趣的读者, 可以参考 Jacquin 的论文 [4]。

^① 文献 [34] 指出, 等距变换的引入徒增了计算复杂性, 因为相同甚至更好的图像质量可以通过减小生成 D 块的步长来达到。但是我们认为, 一方面, 小步长将产生巨大的 D 块池, 于是有更多的 s 和 o 需要计算, 这同样增加了计算开销; 此外, 需要更多的比特数来指明 D 块的位置, 这减少了压缩比。另一方面, 图 7-16 显示了引入等距变换的合理性。因此, 分形编码文献大都考虑了八个等距变换。

另一种实用的方法是忽略问题 (7.5.13) 中内层极小化问题对因子 s 的约束 $|s| < 1$, 从而这个内层约束极小化问题变成标准最小平方问题^[35]:

$$\min_{s, o \in \mathbb{R}} f(s, o) = \min_{s, o \in \mathbb{R}} \|R_i - (s \cdot D_j + o \cdot 1)\|^2. \quad (7.5.14)$$

于是可以采用最小二乘法求解. 这种做法的合理性由熟知实验结果所验证: 对许多标准测试图像的实际计算表明, s 的值绝大多数分布在 $[-1, 1]$, 并在 0 的附近形成峰值或多峰值. 于是, 对于少数不满足约束的 s 作切断处理 (如让 $s = 0$), 图像质量的损失不会太大.

注意到函数 $f(s, o)$ 是变量 s, o 的多元多项式, 它关于 s, o 求偏导数, 并让偏导数等于 0, 得到关于 s, o 的线性方程组. 求解这个线性方程组, 得到最小平方问题 (7.5.14) 的解为 (为简单计, 略去下标)

$$o = \bar{R} - s \cdot \bar{D}, \quad s = \langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle / \|D - \bar{D} \cdot 1\|^2, \quad (7.5.15)$$

其中, 如果 $\|D - \bar{D} \cdot 1\| = 0$, 则 $s = 0, o = \bar{R}$. 函数 $f(s, o)$ 的最小值为

$$\min_{s, o \in \mathbb{R}} f(s, o) = \|R - \bar{R} \cdot 1\|^2 - s^2 \|D - \bar{D} \cdot 1\|^2, \quad (7.5.16)$$

其中, $\langle \cdot, \cdot \rangle$ 表示欧氏内积:

$$\langle R, D \rangle = \sum_{i,j=1}^B r_{i,j} \cdot d_{i,j}. \quad (7.5.17)$$

最小平方问题 (7.5.14) 的求解也可以按向量分量方式进行 (为简单计, 略去下标): 给定 $R, D \in \mathbb{R}^{B \times B}$, 用某种方式 (如行排序) 把它们写成两个向量:

$$R = (r_1, r_2, \dots, r_n)^T, \quad D = (d_1, d_2, \dots, d_n)^T \in \mathbb{R}^n \quad (n = B^2).$$

函数 $f(s, o)$ 可以表达为下面的形式:

$$f(s, o) = \sum_{i=1}^n (r_i - s \cdot d_i - o)^2, \quad (7.5.18)$$

它关于 s, o 求偏导数, 并让偏导数等于 0, 得到关于 s, o 的线性方程组. 求解这个线性方程组, 得到

$$s = \frac{n \sum_{i=1}^n d_i r_i - \left(\sum_{i=1}^n d_i \right) \left(\sum_{i=1}^n r_i \right)}{n \sum_{i=1}^n d_i^2 - \left(\sum_{i=1}^n d_i \right)^2}, \quad (7.5.19)$$

$$o = \frac{\left(\sum_{i=1}^n r_i\right)\left(\sum_{i=1}^n d_i^2\right) - \left(\sum_{i=1}^n d_i\right)\left(\sum_{i=1}^n d_i r_i\right)}{n \sum_{i=1}^n d_i^2 - \left(\sum_{i=1}^n d_i\right)^2} = \frac{1}{n} \left(\sum_{i=1}^n r_i - s \sum_{i=1}^n d_i\right). \quad (7.5.20)$$

对于上述 s, o , 最小误差为

$$\min_{s, o \in \mathbb{R}} f(s, o) = \frac{1}{n} \left[\sum_{i=1}^n r_i^2 + s \left(s \sum_{i=1}^n d_i^2 - 2 \sum_{i=1}^n d_i r_i + 2 \cdot o \sum_{i=1}^n d_i \right) + o \cdot \left(o \cdot n - 2 \sum_{i=1}^n r_i \right) \right], \quad (7.5.21)$$

在上面的式子中, 如果式 (7.5.19) 的分母为零, 则 $s = 0, o = \sum_{i=1}^n r_i / n$.

接着把上面得到两个实数 s, o 代入式 (7.5.13), 问题 (7.5.13) 的外层极小化问题则采用合适的搜索方法求解. 最基本的方法是全搜索, 可以得到较好的编码图像质量, 然而它需要较长的时间, 可见这不是一种实用的方法. 因此, 在保持图像质量基本不变的前提下尽量减少搜索量, 是分形编码技术实用化研究的关键问题, 构成了分形编码的一个主要研究方向.

最后是参数的量化与比特分配问题. 每个 R_i 的分形码涉及四个参数 $\{k, m(i), s_i, o_i\}$, 前两个参数 $k, m(i)$ 分别指明等距变换和最佳匹配块的序号, 是不能量化的. 因此, 只有后两个参数才需要量化. 一般采用一致量器 (uniform quantizer) 来量化参数 s, o , 后面的实验表明, 分别用 5 比特、7 比特量化参数 s 和 o 是合适的. 此外, 和其他图像编码方法一样, 为了进一步提高压缩比, 可以采用考虑参数概率分布的熵编码对量化后的 s, o 进行二次编码.

(六) 解码过程

解码是相对简单的迭代过程, 重构图像是分形码描述的压缩变换 T 的近似不动点图像, 由变换 T 迭代作用于任何初始图像来生成:

$$T^N \mu_0 \approx \lim_{n \rightarrow \infty} T^n \mu_0 = \mu_{\text{fix}},$$

其中 μ_0 是与原始图像同尺寸的任意初始图像, μ_{fix} 是压缩变换 T 的不动点图像, N 是预设的迭代次数. Banach 不动点定理保证解码序列 $\{T^n \mu_0\}$ 是收敛的, 拼贴定理保证其极限 (变换不动点图像) 是原始图像的一个近似: $\mu_{\text{fix}} \approx \mu_{\text{org}}$. 具体地说, 在分形解码过程中, 第 k 次迭代图像 $\mu^{(k)} = T\mu^{(k-1)}$ 的 R 块由下面的迭代方程确定 (未考虑等距变换):

$$R_i^{(k)} = s_i \cdot D_{m(i)}^{(k-1)} + o_i \cdot \mathbf{1}, \quad D_{m(i)}^{(0)} = D_{m(i)}, \quad (7.5.22)$$

其中, $D_{m(i)}^{(k-1)}$ 是 $R_i^{(k)}$ 的最佳匹配块, 源于第 $k-1$ 次迭代图像 $\mu^{(k-1)}$, $D_{m(i)}$ 是源于初始图像的 D 块; s_i, o_i 是 $R_i^{(k)}$ 对应的最佳对比度和亮度的调整系数. 接着按

照分形码提供的分割信息拼贴出图像 $\mu^{(k)} = \bigcup_i R_i^{(k)}$. 如果 k 达到预先设定的迭代次数 N (如 10), 则迭代停止, 并输出图像 $\mu^{(N)}$, 这就是重构图像, 即原始图像的近似.

由此不难看出, 分形解码中每次迭代图像所需要的码本都是由上一次迭代图像提供的, 而第一次迭代图像所需要的码本则由初始图像提供. 这是分形编码不同于 VQ 编码的地方. 此外, 新颖的迭代解码方式也是分形编码不同于其他编码方法之处.

(七) 编码 / 解码算法的具体步骤

基于上述描述, 一个基本分形编码 / 解码算法的具体步骤如下:

1. 编码算法

Step 1 分割图像: 把原始图像按分辨率 $B \times B$ 像素 (如 4×4) 分割成若干块 (记为 R_i), 构成 R 块池. 相邻 R 块之间没有重叠, 它们的并集刚好为原图像.

Step 2 构成码本: 在原始图像中按步长 δ (如 $\delta = 4$) 从左到右、自上而下滑动一个 $2B \times 2B$ 窗口得到 D 块池. 每个 D 块经 4 邻域像素值平均 (或欠采样) 得到 $B \times B$ 像素块, 这种子块的全体就构成码本 Ω .

Step 3 获取分形码: 对于每个 R 块 R_i , 可以按下面三个步骤在码本 Ω 中寻找其最好匹配块 $D_{m(i)}$:

Step 3.1 对于每个 $D \in \Omega$, 按下面的步骤计算参数 s 和 o 的最佳值:

Step 3.1.1 按最小二乘法得到的公式 (7.5.19) 和 (7.5.20) 计算参数 s_i 和 o_i .

Step 3.1.2 分别用 $s_{\text{bit}}, o_{\text{bit}}$ 比特量化参数 s_i, o_i , 例如采用一致量化器.

Step 3.1.3 用参数 s_i, o_i 量化值计算 (7.5.21) 定义的误差, 记为 $E(R_i, D)$, 在码本 Ω 中寻找误差最小的 $D_{m(i)}: E(R_i, D_{m(i)}) = \min_{D \in \Omega} E(R_i, D)$.

Step 3.2 $D_{m(i)}$ 经 8 个等距变换扩大为 8 个子块 $t_j(D_{m(i)})$, 求出误差最小的 $k: E(R_i, t_k(D_{m(i)})) = \min_{0 \leq j \leq 7} E(R_i, t_j(D_{m(i)}))$. 于是得到 R_i 的最好逼近 $s_i \cdot t_k(D_{m(i)}) + o_i \cdot 1$.

Step 3.3 输出当前 R 块 R_i 的分形码, 即量化参数 s_i, o_i 、最匹配码字的下标 $m(i)$ 以及等距变换的序号 k .

Step 4 编码每个 R 块: 对于每个 R 块 R , 重复步骤 3 直至所有 R 块为止.

Step 5 输出经量化后的分形码, 即得到了分形编码文件.

说明:

(1) 因为用最小二乘法得到的参数 s 未必满足约束, 一般把不满足约束的 s 作切断处理 (如取为 0), 或者用次优匹配块代替, 直到找到符合条件的匹配块.

(2) 上述基本分形编码器输出的分形码仅仅是一个图像算子 T 的描述, 它只是告诉接收端 (解码器) 图像算子 T 是如何构成的。解码器必须按照码给出的描述构造出图像算子 T , 然后迭代作用于任意的初始图像 $\mu_0: \mu^{(1)} = T(\mu_0), \mu^{(k)} = T(\mu^{(k-1)})$ (迭代解码)。实验证明 (见下面), 7、8 次迭代即可得到稳定图像 (原图像的近似)。这种迭代解码方式是分形编码有别于其他编码的新颖之处。

(3) 因为等距变换不过是重排图像块的亮度值, 因此, 它们可以看成是另一种亮度变换。于是, 等距变换可以与仿射亮度变换合并为一个新的亮度变换, D 在这个亮度变换下的像为 $s \cdot t_k(D) + o \cdot 1$ 。在这种情况下, 新码本 Ω 则由原码本的 8 个等距变换版本组成, 上述算法中第 3 步需要作微小修改。

2. 解码算法

Step 1 从分形编码文件中读取 R 块、D 块、步长以及原图尺寸的信息。

Step 2 根据所得的数据, 定义两个和原图一样大小的图像区: R 区和 D 区。R 区用来保存迭代过程中所生成的图像, D 区用来产生码本 Ω 。

Step 3 初始化 D 区图像。

Step 4 对于 R 区中每个 R 块 R_i , 根据分形文件中的分形码 $\{k, m(i), s_i, o_i\}$, 在 D 区中找到相应的 D 块 $D_{m(i)}$, 则 $R_i = s_i \cdot (t_k(S(D_{m(i)}))) + o_i \cdot 1$ 。得到所有的 R 块后, 经拼贴后就形成了一次迭代的图像。

Step 5 将 R 区的图像复制到 D 区。

Step 6 如果迭代的次数达到预定次数, 则停止。否则转到第 4 步。

Step 7 输出重构图像。

上述编码 / 解码算法的程序代码 (编码 / 解码的主体程序函数), 见本书附录。程序用 C++ 编写, 并在 C++ builder 里编译通过。本程序假定用 5 bit 存储 s , 用 7 bit 存储 o , 用 16 bit 存储 D 块坐标, 用 3 bit 存储等距变换的方式。

四、实验结果与结论

在上述基本算法中, 重构图像质量、编码时间和压缩比依赖于许多因素, 如 R 块、D 块的大小、生成 D 块池的步长、是否采用等距变换、亮度调整因子和亮度偏移量的量化比特数, 等等。

在实验中, 我们选取 R 块大小为 44×4 , D 块大小为 8×8 (即 $B = 4$), 并采用 8 个等距变换, 同时用 5 bit 存储亮度调整因子 s 、用 7 bit 存储亮度偏移量 o 。此外, 因为 D 块的尺寸选为 R 块尺寸的 (边长) 两倍, D 块的位置可以由相邻 D 块之间的间隔 δ (步长) 决定。 δ 可以小到一个像素 ($\delta = 1$), 也可以大到 $2B$ ($\delta = 2B$)。选取 $\delta = 1$ 的方案将产生十分庞大的 D 块池, 这会导致后来的搜索非常耗时。本实验选取 $\delta = 2B$, 即 $\delta = 8$ 。

实验在 AMD Duron/850MH、运行 Windows 2000 的计算机上进行。测试编解

码性能的参数是峰值信噪比 (PSNR, 单位: dB) 和编码时间 (单位: 秒, 源于操作系统时钟). 峰值信噪比定义见 (7.5.11).

我们选择标准测试灰度图像 256×256 Lena 图像 (8bit 量化) 作为实验对象, 考虑到随机因素和运行环境等因素的影响, 我们对这幅图像进行了 10 次编码, 结果表明平均编码时间约 130 秒, 可见用基本分形算法编码图像是十分费时的.

相比分形编码, 分形解码却非常快 (见图 7-20~图 7-22). 因此, 分形编码技术较适用于一次编码多次解码的应用中.

图 7-20 给出了标准测试图像 “Lena” 的前 5 幅解码图像序列 (初始图像 μ_0 是全黑图): 从左到右、自上而下分别是初始图像 (全黑图)、前 5 次迭代的解码图, 即 $T^1(\mu_0)$ 、 $T^2(\mu_0)$ 、 $T^3(\mu_0)$ 、 $T^4(\mu_0)$ 、 $T^5(\mu_0)$.



图 7-20 初始图像为全黑图时前 5 次迭代的解码序列

图 7-21 给出了初始图像 μ_0 是 “bird” 图像时 “Lena” 图像的前 5 幅解码图像序列: 从左到右、自上而下分别是初始图像 “bird”、前 5 次迭代的解码图像, 即 $T^1(\mu_0)$ 、 $T^2(\mu_0)$ 、 $T^3(\mu_0)$ 、 $T^4(\mu_0)$ 、 $T^5(\mu_0)$.

图 7-20 和图 7-21 显示, 分形图像解码速度是很快的, 第一次迭代解码图像已完全没有初始图像的踪影 (图 7-21), 且从任意初始图像 (如全黑图和 “bird”) 迭代 4 次就能得到原图像的较好近似, 10 次迭代就得到稳定图像 (图 7-22).

此外, 虽然分形解码与初始图像的选择无关, 但解码速度与初始图像的选择有关. 比较图 7-20 和图 7-21, 容易看出初始图像为全黑图时的解码速度快于初始图像

为“bird”图像时的解码速度。

图 7-22 定量显示了初始图像为全黑图像时的 Lena 图像的解码收敛情况。从图中可以看出, 10 迭代的解码图像序列已基本收敛, 或者说 10 次迭代的解码图像质量 (以 PSNR 度量) 已基本稳定: 对应于迭代次数 10, 15, 20 和 50, 解码图像的 PSNR 分别约为 27.78dB, 27.827dB, 27.829dB 和 27.829dB, 第 10 次迭代的解码



图 7-21 初始图像为 bird 图像时前 5 次迭代的解码序列

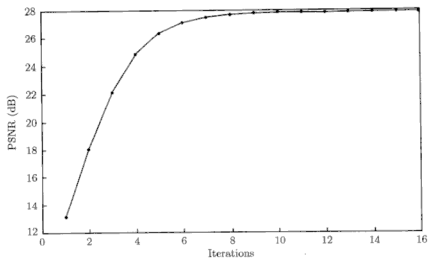


图 7-22 初始图像为全黑图时 Lena 图像的解码收敛图

图与更多次迭代的解码图的 PSNR 仅仅相差不到 0.05dB. 因此, 在 0.05dB 的误差范围内, 第 10 次迭代解码图像就已达到解码图像序列的收敛图像.

第六节 本章小结

本章从迭代函数系统的观点介绍了分形编码基本类型的原理, 并从 VQ 观点介绍了基本算法及实现. 需要指出, 本章介绍的分形算法是分形图像编码中最基本、最简单的形式, 设计分形压缩软件时是不会使用这种简单形式的. 但它有简单易懂、又能显示分形编码实质的优点, 是介绍分形编码算法的最好教材, 在许多分形编码文献中也常常用来显示新算法的改进之处.

从实用角度看, 上述基本分形编码算法的计算复杂性很高, 编码质量也不理想. 如何减少编码时间、提高编码质量, 还有许多待解决的公开问题, 这些问题为应用数学、电子工程和计算机科学等领域的研究者提供了大展宏图的表演舞台. 例如, 图像如何分割? 如何匹配图像块? 亮度变换如何设计? 等等. 打个形象的比喻, 上述分形编码器好比第一台发动机 (分形图像编码 / 解码系统), 如何用最优方式设计这样的发动机? 这涉及方方面面的问题. 我们将在下一章讨论分形图像编码的各种改进算法, 包括作者的主要工作.

参 考 文 献

- [1] Torres L, Kunt M. Video coding: the second generation approach, Kluwer Academic Publisher, 1996
- [2] Barnsley M, Sloan A. A better way to compress images. BYTE, 1988, 1: 215~223
- [3] Barnsley M, Hurd L. Fractal Image Compression. AK Peters, Wellesley, 1993
- [4] Jacquin A E. Image coding based on a fractal theory of iterated contractive image transformations. IEEE Transactions on Image Processing, 1992, 1(1): 18~30
- [5] Barnsley M. Fractal Image Compression. Notice of the AMS, 1996, 6: 657~662
- [6] Wohlberg B, de Jager G. A Review of the Fractal Image Coding Literature. IEEE Transactions on Image Processing, 1999, 8(12): 1716~1729
- [7] Saupe D, Hamzaoui R, Hartenstein H. Fractal image compression-An introductory overview. in: Fractal Models for Image Synthesis, Compression, and Analysis, Saupe D, Hart J (eds.), ACM SIGGRAPH'96 Course Notes
- [8] 张宏基. 信源编码 (修订本), 北京: 人民邮电出版社, 1987
- [9] Gersho A, Gray R. Vector Quantization and Signal Compression. Boston: Kluwer Academic Publishers, 1992
- [10] Gray R M, Neuhoff D L. Quantization. IEEE Trans. Information Theory, 1998, 44(6): 2325~2383

- [11] Heckbert P. Color image quantization for frame buffer display. *ACM Trans. Comput. Gr.*, 1982, 16(3): 297~307
- [12] Linde Y, Buzo A, Gray R. An Algorithm for Vector Quantizer Design. *IEEE Trans. Commun.*, 1980, 28(1): 84~95
- [13] Murakami T, Asai K, Yamazaki E. Vector quantizer of video signals. *IEE Electronics Letters*, 1982, 7: 1005~1006
- [14] Barnsley M. *Fractals Everywhere*. San Diego: Academic Press, CA, USA, 1988
- [15] Barnsley M, Rising III H. *Fractals Everywhere* (2nd ed.), Boston, MA: Academic Press Professional, 1993
- [16] Peitgen H O, Saupe D (Eds.) . *The Science of Fractals*. New York: Springer Verlag, 1989
- [17] Peitgen H O. *Fractal in the fundamental and applied science*. Elsevier Science Publisher, North Holland, 1991
- [18] Hepting D. Rendering Methods for Iterated Function Systems, in [17], 183~244
- [19] Kominek J. Advances in fractal compression for multimedia applications. *Multimedia Systems*, 1997, 5: 255~270
- [20] Hartenstein H, Saupe D. Lossless acceleration of fractal image encoding via the fast Fourier transform. *Signal Processing: Image Communication*, 2000, 16: 383~394
- [21] Arneodo A, Bacry E, Muzy J F. Solving the inverse fractal problem from wavelet analysis, *Europhysics Letters*, 1994, 25(3): 479~484
- [22] Abenda S. Inverse problem for one-dimensional fractal measures via iterated function systems and the moment method. *Inverse Problems*, 1990, 6(12): 885~896
- [23] Abenda S, Demko S, Turchetti G. Local moments and inverse problem for fractal measures. *Inverse Problems*, 1992, 8(10): 739~750
- [24] Handy C R, Mantica G. Inverse problems in fractal construction: Moment method solution. *Physica D*, 1990, 43(5): 17~36
- [25] Forte B, Vrscay E R. Solving the inverse problem for function and image approximation using iterated function systems: I. Theoretical Basis ; II. Algorithm and computations, *Fractals*, 1994, 2: 325~334 (I) ; 335~346 (II)
- [26] Mitra S K, Sarbadhikari S N. Iterative function system and genetic algorithm based eeg compression. *Med. Eng. Phy.*, 1997, 19 (7): 605~617
- [27] Mas Ribes J M, Simon B, Macq B. Combined Kohonen neural networks and discrete cosine transform method for iterated transformation theory. *Signal Processing: Image Communication*, 2001, 16: 643~656
- [28] Rinaldo R, Zakhor A. Inverse and approximation problem for two-dimensional fractal sets. *IEEE Transactions on Image Processing*, 1994, 3: 802~820
- [29] Cabrelli C A, Forte B, Molter U M, Vrscay E R. Iterated fuzzy set systems: A new approach to the inverse problem for fractals and other sets. *Journal of Mathematical Analysis and Applications*, 1992, 171(15): 79~100

- [30] Cabrelli C A, Falsetti M C, Molter U M. Fractal Block-Coding: A Functional Approach for Image and Signal Processing. *Computers and Mathematics with Applications*, 2002, 44: 1183~1200
- [31] Nettleton D J, Garigliano R. Evolutionary algorithms and a fractal inverse problem. *Biosystems*, 1994, 33(3): 221~231
- [32] Chen J H, Kalbfleisch J D. Inverse problems in fractal construction: Hellinger distance method. *Journal of the Royal Statistical Society, Series B Methodological*, 1994, 56(4): 687~700
- [33] Ruhl M, Hartenstein H. Optimal fractal coding is NP-Hard. in *Proceedings DCC'97 Data Compression Conference*, Storer J A, Cohn M (Eds.). IEEE Computer Society Press, March 1997
- [34] Saupe D. The futility of square isometries in fractal image compression. In *Proc. ICIP-96 IEEE International Conference on Image Processing*, Lausanne, Sept. 1996
- [35] Fisher Y (Ed.). *Fractal Image Compression: Theory and Application*. New York: Springer-Verlag, 1994

第八章 分形编码的改进算法

第一节 引言

我们在上一章详细介绍了分形编码基本类型的原理、算法与实现。从基本原理上讲,分形图像编码的确是十分简单的。编码阶段,一幅图像用一个使图像近似不变的压缩变换的参数来表达,而表达压缩变换参数的比特数大大低于储存待编码图像的比特数,因而实现了图像数据的高倍压缩。解码是新颖的快速迭代过程,Banach 不动点定理保证表达图像的压缩变换的不动点图像是存在唯一的,且可以从任何初始图像迭代生成,拼贴定理保证这个不动点图像是待编码图像的一个近似图像。然而,如何寻求一个不动点图像是待编码图像较好逼近的压缩变换?这是分形图像编码技术最基本的问题。

在解决上述基本问题方面,上一章介绍的基本分形块编码算法实现了计算机自动编码过程,但从实用角度看,基本算法的计算复杂性很高,编码质量也不理想。如何减少编码时间、提高编码质量,还有许多待解决的公开问题:例如,图像如何分割?如何匹配图像块?亮度变换如何设计?等等。打个形象的比喻,基本分形编码器好比造出的第一辆汽车。但是,如何用最优方式设计这样的汽车,还有待于进一步的研究。因此,分形编码算法的具体实现还留有巨大的活动余地。这些问题为应用数学、电子工程和计算机科学等领域的研究者提供了大展宏图的表演舞台。

纵观大量分形图像编码文献,在上述基本原理的框架内,在提高编码质量和加快分形编码等方面,许多新观点和大量改进算法被提出,对它们进行准确的分类是困难的。但总的来说,大多数现有分形编码文献都是围绕下面的基本问题展开的,或者说大多数分形编码方案的区别主要反映在下列几方面:

(1) 图像分割。尽管固定方块分割方案实现简单,也有分割方案占有零信息量的优点(即无需存储或传输分割信息),但编码质量不理想,因为方块尺寸必须事先指定且不依赖于待编码图像,也没有利用图像内容,不能自适应于不同的图像,也不能自适应于同一图像的不同细节(如飞行中的飞机照片,飞机与蓝天的细节显然不同)。四叉树分割方案虽然有一定的图像适应性,且描述四叉树分割的成本也很小,但它对图像的适应性仍然是有限的。因此,研究比四叉树分割的图像适应性更强的自适应分割方案是一个重要的问题。

(2) 虚拟码本构成。从迭代函数系统观点来看,这就是如何构造 D 块池的问题。与 VQ 编码一样,合适的虚拟码本对编码时间和编码质量都有很大影响。以固

定方块分割为例,增大虚拟码本的容量(如以小步长生成D块池、同时考虑八个等距变换),编码质量自然会提高,但这必须以牺牲编码时间和压缩比为代价,因为虚拟码本越大,匹配搜索的范围越大,编码时间自然越多,同时也需要更多的比特数来指明最佳匹配块的位置.在虚拟码本构成方面,许多VQ码本构成策略被移植到分形图像编码中.

(3) 亮度变换类型.亮度变换是指作用于码本块的变换,基本分形图像编码的亮度变换仅仅实现亮度调整与亮度补偿,即每个R块用码本块和常值亮度块的线性组合来编码($R \approx s \cdot D + o \cdot 1$).显然,按这种方式构造的变换具有结构简单、计算方便的优点,但是其编码能力是有限的.没有理论显示这种选择就是最优的.

(4) 变换参数的量化、比特分配与熵编码.在分形编码中,一幅图像是用使图像近似不变的压缩变换来表达的,而变换又是由表达它的参数确定的.为了进一步实现图像数据的压缩,表达变换的亮度调整参数和亮度偏移参数必须进行量化处理,并对量化参数进行合理的比特分配(bit allocation).此外,熵编码可进一步改进分形编码的效率.

(5) 分形解码问题.尽管解码快是分形图像编码公认优点之一,但在实时性要求高的应用领域(如实时视频),更快的解码是必需的.于是更快速分形解码得到人们的关注.另一方面,在另外一些应用领域(如制作动画、窄带传输),需要能够以某种方式控制解码过程.对于后一方面,目前只有作者为此进行了探索,并首次引入“渐进/可控分形解码”的概念,提出了一个新颖的分形解码算法,部分实现了可控分形解码.

(6) 基础理论研究.迭代函数系统逆问题的研究是其核心问题,尽管人们已尝试各种方法解决这个问题,但目前仍然没有得到理想解决.我们知道,压缩图像的分形文件就是一个压缩变换的描述,对于如何寻求这样的压缩变换,拼贴定理是目前依据的理论基础.但是,拼贴定理只是给重构误差提供一个上界(体现为所谓的拼贴误差),这个上界极小化并不意味着重构误差的极小化.因此,目前的分形编码方案远不是最优的.当然,相关的基础理论问题并不止这个逆问题,例如,更好的数学框架、解码收敛性、解码误差控制以及分形编码机理等的研究.

(7) 加快编码过程.这主要涉及匹配搜索问题,因为在分形编码阶段,匹配搜索占据了主要的编码时间.全搜索固然能够得到最好的编码质量,但全搜索的运算量大是不争的事实.加快分形编码速度的关键是设计好的搜索方案(与虚拟码本的构成密切相关),这是分形图像编码实用化的核心问题.

(8) 混合分形编码.大多数纯粹基于分形理论的编码方法与目前广泛使用的编码方法相比,还缺乏竞争力,但是,分形编码与其他编码方法(如VQ、小波、神经网络等)结合的混合编码方法已经取得巨大成功.

应该指出,上述清单并没有涵盖分形图像编码的全部内容.此外,上述各方面是密切相关的,比较某一方面的不同决策的编码性能,常常需要考虑其他方面的相

应决策。例如，加快编码过程与上述其余的部分都紧密相关，虚拟码本（或 D 块池）搜索方案在某种程度上取决于虚拟码本的构成，等等。遗憾的是，几乎没有理论结果能指导我们做出相应决策，只能凭经验或仿真试验。

针对上述各个方面，目前已提出的编码方案非常多，囿于篇幅和作者水平，我们仅仅介绍其中部分内容并列出部分参考文献。

第二节 图像分割

分形图像编码的基本思想，就是寻找图像不同区域之间在不同尺度下的相似性。因此，与通常的图像编码方法一样，设计分形编码系统的第一步就是进行图像分割，即把待编码图像划分成若干个子区域（R 块），每个子区域对应于图像中的某一物体或物体的一部分，接下来的主要步骤就是对每个子区域寻找与其仿射相似的大区域（D 块）。这样，每个 R 块都对应一组仿射变换系数，若不计分割信息并用等长码编码系数，则分形码的文件长度正比于 R 块的数目。因此，分割是决定压缩比的关键因素。

分割也是决定解码图像质量的又一关键因素，一个好的分割方案应该反映出图像的跨尺度相似性。图像中既有平滑均匀区域（亮度恒定或缓慢变化的区域），又有高对比度区域（如边缘区域）。在均匀区域部分，使用大块就能实现好的拼贴，与此同时，高对比区域则需要使用小尺寸块才有可能达到希望的图像质量。要实现这一点，必须采用更为灵活的分割方法（可变尺寸块分割）。包含四叉树（quadtree）分割在内的各种可变尺寸块分割方法（自适应分割）已经在分形编码文献中得到广泛研究。

分形码（一个量化压缩仿射变换的描述）是由分割信息与量化变换参数组成的，不同分割方案占有的分割信息量的大小是不同的。从压缩观点看，分割方案占有的信息量越少越好，但是，如果一个不规则分割带来了好的主客观编码质量，花费一点编码成本是值得的。因此，在影响分形图像编码性能的诸多因素中，分割应该是最重要的因素之一。选择何种分割方案需要权衡压缩比和编码质量并选择一个折中的方案。

本节简单介绍目前已提出的多种图像分割方案。

一、固定尺寸块

在分形图像编码中，最基本的一类分割是固定尺寸块分割或均匀分割（uniform partition），分割由形状、尺寸固定的图像块构成（如方块、三角形块、矩形块等）。固定方块分割是最基本的均匀分割，分割由固定尺寸（如 8×8 ）的方块组成（如图 8-1a 所示）。这种固定方块分割首次出现在 Jacquin 提出的分形块编码算法^[1]中。其他

类型的均匀分割可以由平面的不重叠的规则覆盖来定义,但很少在分形编码中使用。

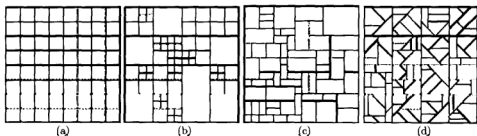


图 8-1 四种分割: (a) 固定方块; (b) 四叉树; (c) HV; (d) 多边形

固定方块分割在单个图像块的变换编码中是成功的 (正如 JPEG 标准^[2]实现的那样), 因为自适应的量化机制能够对不同块的“活性”程度作出补偿, 细节少的子块分配较少的比特, 细节丰富的块则给予较多的比特。但是, 基于块变换的分形编码则没有这样的后续补偿适应能力^[3]。对于基于方块的均匀分割 (设子块大小为 $B \times B$), 块尺寸 B 必须先指定且不依赖于待编码图像, 因此它没有利用图像内容, 不能自适应于不同的图像, 也不能自适应于同一图像的不同细节 (如飞行中的飞机照片, 飞机与蓝天的细节显然不同)。我们在实验中发现, 若采用固定方块分割, 在一些典型图像中, 一些 R 块在给定的失真容忍度下不可能找到与之匹配的 D 块, 只好用次优块代替。这是固定方块分割的显著缺点。

固定方块分割也有其优点, 例如分割方案占有的信息量为零, 分形码仅由变换参数组成, 无需存储或传输分割信息。特别是固定方块分割理解容易、实现方便, 使它常常出现在分形编码文献中, 用于介绍分形编码原理、说明新算法的改进之处等等。

二、层次分割

从编码质量方面看, 图像的适应分割比固定尺寸块分割有更大的优越性。一般图像中既有平滑均匀区域, 又有高对比度区域 (如边缘附近)。为了达到希望的压缩比又能得到好的解压图像质量, 在均匀区域部分, 可使用较大的块, 在高对比区域则使用较小的块。首次提出实用分形编码算法的 Jacquin^[1] 开始就已注意到这个事实, 为了改进图像质量, 他在其算法中考虑了可变尺寸的方块, 例如 4×4 , 8×8 , 16×16 方块。他的想法导致了在分形编码中使用四叉树分割的方案 (始于文献 [4]), 以及后来提出的各种自适应分割方案, 它们都可以归入 (自顶而下或自底而上的) 层次分割的范畴。

与固定尺寸块分割编码不同, 各种自适应分割编码的输出文件必须包含分割

信息(牺牲一点压缩比),但可变尺寸分割使设计可变数码率编码器有了可能,用户多了一个选择,可以根据需要选择图像质量或压缩比。

(一) 四叉树分割

四叉树(quadtrees)是一棵每个非叶子结点都有4个子结点的树。四叉树分解就是用一棵四叉树表示一幅图像,树根是原始图像,树的每个结点(除树根外)对应一个图像块,这个图像块又是其父块的四等分子块之一。换言之,在图像的四叉树表示中,整个图像用树根联系,然后将该图像等分成4个子图,它们联系于四叉树的4个子结点。图8-2(a)给出了四叉树分割的一个例子,图8-2(b)是对应的四叉树表示。

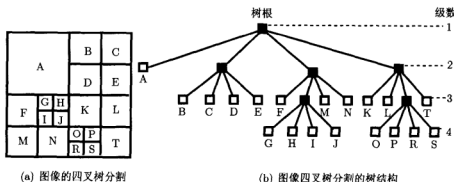


图 8-2 图像四叉树分割与四叉树表示的示例

描述四叉树结构的成本是很小的。例如,为了储存或传输四叉树结构,一个简单的方法是通过深度优先搜索法(depth-first search)遍历这棵树。若一个叶结点被访问则标记“1”,内部结点标记“0”,则图8-2所示的四叉树结构的编码为

01011101011111011011111.

尽管描述四叉树分割的成本不高,但基于四叉树的分割对图像的适应性仍然是有限的。

四叉树分割方案可以看成是向适应分割迈出的第一步,其中基于自顶而下四叉树的分形编码器始见于文献[4]。开始把整幅图像看成一个单一的父块(即四叉树的树根),当拼贴误差超过预定阈值时,图像被等分成四个子块(四叉树树根的四子结点)。依次考虑四个子块的每一块(新的父块),当拼贴误差超过预定阈值时,新父块又被等分成四个子块。继续这个过程直至中止条件满足(或均匀性检测准则为真)。文献[5]的研究表明,基于拼贴误差的分裂准则可以代之与R块的灰度方差准则(方差要乘以R块的像素数目),率失真(rate-distortion)性能几乎不受影响,从而在编码性能几乎不变的条件下加快了编码过程。文献[6]提出了适应阈值的四叉树分割方案,在相同条件下与固定阈值的四叉树分割相比,编码质量明显好于

后者。他们的做法是, 设第 i 层的阈值为 e_i , 下一层 ($i+1$ 层) 的阈值由 $e_{i+1}=ke_i$ 确定 (他们取 $k=2$)。因此, 如果第一层的初始阈值选定为 e_1 , 则第 i 层的阈值为 $e_i = k^{i-1}e_1$ 。显然, $k=1$ 对应固定阈值四叉树分割, $k=2$ 对应文献 [6] 的适应阈值四叉树分割。率失真最优四叉树分割能够以自底向上的方式得到 [7], 并被许多分形压缩文献采用 (如文献 [8~10] 等)。

尽管描述四叉树分割的成本不高, 但四叉树分割对图像的适应性仍然是有限的。一个较高适应四叉树分割可以通过融入分裂后合并的策略 (图像被等分成四个子块后考虑子块的合并) 得到, 文献 [11] 讨论了这个方法。

(二) HV 分割

基于 HV 分割 (HV 是 horizontal-vertical 的缩写) 的分形编码在文献 [12] 中提出。与四叉树分割一样, HV 分割也产生图像的一个树状结构, 不同的是图像被分割成矩形块。对于一个给定的矩形块 (R), 如果具有可接受拼贴误差的匹配块 (D) 找不到, 则该矩形块 (R) 用水平线或垂直线分裂为两个矩形块 (图 8-1c)。分裂基于最显著的水平边缘或垂直边缘进行, 也融入了矩形退化保护机制。具体地说, 对于一个矩形块 $R = (r_{i,j})_{N \times M}$, 垂直方向像素亮度值之和的偏差 h_j 与水平方向像素亮度值之和的偏差 v_i 分别由下面的公式计算:

$$h_j = \frac{\min(j, M-j-1)}{M-1} \left(\sum_{i=0}^{N-1} r_{i,j} - \sum_{i=0}^{N-1} r_{i,j+1} \right), \quad 0 \leq j \leq M-1, \quad (8.2.1)$$

$$v_i = \frac{\min(i, N-i-1)}{N-1} \left(\sum_{j=0}^{M-1} r_{i,j} - \sum_{j=0}^{M-1} r_{i+1,j} \right), \quad 0 \leq i \leq N-1, \quad (8.2.2)$$

这些差值 (绝对值) 的最大值决定了分裂的方向和位置。包含这个信息的决策树必须被储存或传输。图 8-3 给出了一个示例。

对于每次分裂, 分裂的方向和位置都要被编码, 因此 HV 分割一般比四叉树分割需要花费更多的分割信息记录成本。尽管如此, 仿真实验结果表明, 基于 HV 分割的分形编码器的性能大大优于基于四叉树分割的分形编码器 [12]。

尽管 HV 分割比四叉树分割进了一步, 但 HV 分割对图像的适应性仍然是有限的, 因为图像局部区域之间的相似性未必都落在长方形内 (率失真最优 HV 分割的详细介绍请看参考文献 [13])。

对图像适应性的进一步改进是采用基于文献 [14] 提出的图像分割方法 (一种多边形分割), 被 Reusens^[15] 首次应用于分形图像编码。它类似于 HV 分割, 但它容许 45° 线和 135° 线 (相当于正方形的次对角线与主对角线) 分裂。Reusens 使用拼贴误差作为分裂准则, 分裂方向和位置由基于方差的块均匀性检测准则确定。顺便指出, Reusens 是为分析高图像适应性能否改进分形编码方案的率失真行为而

研究上述分割方法的。他比较了基于这种分割的编码器、基于二叉树分割的编码器和基于 HV 分割的编码器 (作为多边形分割的特殊情况对待, 即把分裂限于水平方向与垂直方向) 的性能。他的结论是, 好的适应性分割并不能改进率失真性能 (有些学者的结论与此不同, 见下面的内容)。



图 8-3 一个 HV 分割

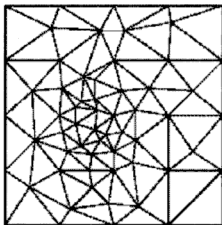


图 8-4 Delaunay 三角剖分

(三) 多边形分割

人们研究了不同的三角形分割方法。一种做法是, 开始时通过插入一条合适的对角线把图像划分为两个大三角形, 这两个大三角形可以通过两种方式之一细分为小三角形。一种方式是采用三边分裂 (3-side split)^[16], 在需要分裂的三角形的三条边上, 各选取一点作为三角形的新顶点; 另一种方式是采用一边分裂 (1-side split)^[17], 在需要分裂的三角形的一个顶点与对边之间插入一条直线把三角形分裂为两个小三角形。另一种做法是利用 Delaunay 三角剖分, 我们将在下面的分裂 - 合并分割中介绍。

多边形分割是从初始的粗略网格开始, 在网格中可以任意地插入一条线进行递归分割^[15](图 8-1d), 也可以合并三角形得到四边形^[18](见下面的分裂 - 合并分割)。文献 [19] 提出了六边形分割和轮廓线分割, 文献 [20] 提出了菱形分割。我们不再介绍这些方法, 有兴趣的读者可以参看原文。

三、分裂 - 合并分割

分裂 - 合并分割是图像分割技术的一种重要方法。基本原理是利用图像数据的金字塔结构的层次概念, 将图像划分成一组任意不相交的初始区域, 即可以从图

像的这种金字塔数据结构的任一中间层开始, 根据给定的均匀性检测准则进行分裂与合并这些区域, 逐步改善区域划分的性能, 直到最后将图像分成数量最少的均匀区域为止。分形图像编码的基本思想就是寻找图像中的不同区域间的跨尺度相似性, 分形编码中融入这种技术能够改进编码性能是可以预想的。

Davoine 等研究者^[18,21,22]提倡用 Delaunay 三角剖分 (Delaunay triangulation, 图 8-4) 作为分割方法。三角剖分的优点是不受边缘方向的限制。Delaunay 三角剖分是一种最大化三角形最小内角的剖分, 它利用了某种均匀性准则。具体做法如下: 从一组规则分布的点 (可以看成“种子点”) 的 Delaunay 三角剖分开始, 通过分裂不均匀三角形来加细 (refine) 分割 (这里把分割看成 R 块的集合), 所用的均匀测度由 R 块的亮度方差定义。所谓分裂不均匀三角形, 就是把不均匀三角形的重心添入种子点集, 然后重新计算新的点集对应的 Delaunay 三角剖分。分裂过程由均匀测度判断控制。接下来是合并过程, 如果所有以 p 为顶点的三角形具有近似相等的亮度均值, 则去掉顶点 p , 然后重新计算新的点集对应的 Delaunay 三角剖分。在文献 [18, 22] 中, 两个三角形满足下面的条件也容许合并, 即它们构成凸四边形且两个三角形具有大致相同的灰度分布。分割由指定分裂-合并过程来编码。为了处理大量不同的几何形状, Davoine 等使用另外的几何变换代替通常的仿射变换。文献 [22] 指出, 基于这种分割方法的编码结果优于基于 HV 分割的编码结果。

使用启发式搜索的基于区域的分形编码器 (region-based fractal coder using heuristic search) 也可以看成一种分裂-合并方法, 是 Thomas 和 Deravi^[23]提出的。首先把图像进行均匀分割, 即把图像划分成原子方块 (如 4×4 、 8×8 像素), 然后依次合并相邻的方块得到不规则形状的大块 (R)。最后只有少数大块 (R), 因此需要储存或传输的变换参数不多。因为图像块 (R) 大小不一、形状又不规则, 不能使用传统的搜索方法, 必须采用一定的启发式策略。Thomas 和 Deravi 给出了三种复杂性不同的算法, 拿最基本的算法来说, 对每个原子方块搜寻其最优匹配块 (D), 然后依次对每个原子方块检查变换能否延拓到其邻域块上去 (这一延拓过程由失真判断控制), 接着对另一个原子方块施行这一延拓过程, 直至整幅图像被编码。另外两种方法也包含了一些更新手续 (updating procedure) 并内建了竞争机制。

另一种基于进化计算 (evolutionary computation) 的适应分割方法由 Saupe 和 Ruhl^[24]提出。在这种方法中, 对于一个固定方块分割, 分形码按标准分形编码获取, 但是对每个 R 块, d 个匹配最好的码本块连同相应的最佳尺度因子和亮度偏移量要保存在一个链表 (list) 中。然后取 N 个这种构形 (configuration) 作为进化的初始种群。随机合并两个相邻块, 仅仅考虑参与合并的两个相邻块所在链表中的变换来修改分形码, 并从中选择 (拼贴误差意义下) 最佳构形生成下一代 (这定义了选择算子)。Saupe 和 Ruhl 的实验结果表明, 进化编码方法得到了比四分树分割更好的率失真曲线。

四、重叠块

为了消除分形解码图像的块效应,文献[25~27]讨论了重叠块分割方案.文献[25]和文献[26]分别是在四叉树分割和固定块分割中融入块重叠策略的,虽然在消除块效应方面有一定的效果,但以均方误差(MSE)定义的信噪比却没有相应提高.文献[27]虽然是考虑块重叠的固定块分割,但块重叠的形式更为复杂,因此,提高了信噪比和主观质量.

这些方法在消除块效应方面是有希望的,但块重叠部分被重复编码有悖于数据压缩的宗旨,此外效果也远不及小波域上的分形编码方法.我们这样说并非是要否定这种方法,实际上,文献[28]正是利用块重叠策略发展了一种类似于 Fourier 变换、正交函数展开和小波变换的广义分形变换.另外,文献[29]提出的分形块编码的函数方法也容许块重叠,并在二维信号情形(\sqrt{x} 和 $\sin \pi x$)得到很好的重构质量.

五、小结

本节简单介绍了目前提出的多种图像分割方案(也许不止这些).不难发现,以上的分类不是十分准确,因为有些分割方法可以同时跨不同的类.不过我们重在分割方法本身,至于如何归类才准确不是我们的论题.

说到不同分割的优劣,似乎难于定论.例如,文献[15]比较了分别基于多边形分割、HV 分割与四叉树分割的分形编码器的性能,发现最简单的分割(四叉树)提供了最好的率失真结果.因此该文得出结论,好的适应性分割并不能改进率失真性能.与此相反,文献[30]单独比较了基于 HV 分割与四叉树分割的分形编码器,发现 HV 分割优于四叉树分割.此外,有文献报告不规则分割既优于固定块分割[23,30],又优于四叉树分割[24,31].

总之,在影响分形图像编码性能的诸多因素中,分割应该是最重要的因素之一.我们认为,太不规则的分割给算法实现带来麻烦,而且需要大量的额外开销以编码区域的形状和位置.因此,在设计分形压缩软件时,选择何种分割方案需要权衡压缩比和编码质量并选择出一个折中的方案.

第三节 虚拟码本构成

从迭代函数系统观点来看,分形图像编码中构造虚拟码本的问题就是构造 D 块池的问题.我们知道,在 VQ 编码中,构造码本是十分重要的问题.同样,在分形图像编码中,虚拟码本对编码时间和编码质量都有很大影响.一般来说,虚拟码本越大,编码质量越高,但这必须以牺牲编码时间和压缩比为代价.这是因为虚拟

码本越大, 匹配搜索的范围越大, 搜索到最好匹配块的可能性也越大, 但匹配搜索的时间自然越多, 同时也需要更多的比特数来指明最佳匹配块的位置。

在分形图像编码文献中, 依据不同的实验结果, 虚拟码本可以分成全局码本与局部码本两大类。

一、全局码本

在上一章描述的基本分形算法中, 每个 R 块共用同一本固定的虚拟码本。该码本的构成有一个特点, 码本块与 (对应的) R 块之间没有位置上的相邻性。其依据是一些学者的实验结果^[32]: 对于一个给定的 R 块, 其最好匹配 D 块未必是它的近邻 (指位置关系)。

在基本分形算法中, 相邻 D 块之间的间隔可以小到一个像素 (即步长取为 1), 但这样做将产生十分庞大的 D 块池, 从而导致后来的匹配搜索非常耗时, 既减少了压缩比, 解码图像质量增益也不大。因此, 人们通常选取较大的步长, 一般选用两个步长, 一个为 D 块的边长, 另一个为 D 块边长的一半。此外, 如何选取 D 块的尺寸也是值得详细研究的问题^[33], 通常的做法是选取为 R 块尺寸的两倍。但是, 图像算子的压缩性并不需要 D 块的几何收缩性^[33], 因此, D 块的尺寸可以任意选择, 选取为 R 块尺寸的两倍可能是为了实现方便。

二、局部码本

与上述实验结果不同, 许多学者在实验中观察到这样的事实, R 块与其最佳匹配块在几何位置上有相邻的趋势, 即 R 块与其最佳匹配块的几何距离分布在原点处形成峰值^[34]。受此启发, 一些学者提出了局部码本生成策略, 例如, 把 D 块限制在对应的 R 块的附近, 从 R 块的位置出发按螺旋方式向外扩展生成局部码本, 或者其他一些更为复杂的构成方法 (详见综述文献 [3] 中列出的参考文献)。

三、小 结

全局码本与局部码本构成方案依据了不同的实验结果。对于一个给定的 R 块, 一种实验结果指出其最好匹配 D 块在几何支持上未必是它的近邻, 另一种实验结果显示其最佳匹配块在几何位置上有相邻的趋势, 即 R 块与其最佳匹配块的几何距离分布在原点处形成峰值。文献 [3] 认为这两种实验结果是矛盾的。这种看法值得商榷。我们认为前者是从确定意义上说的, 而后者是从统计意义上得出的结论。不难想见, 按照局部码本的解码图像质量肯定不如全局码本编码的图像质量, 但前者节约了编码时间。因此, 我们不能单纯地评价两种码本构成的优劣, 必须综合考虑编码时间和解码图像质量。对于编码速度要求高而图像质量次之的应用场合, 当

然应该首选局部码本, 否则, 应该选择全局码本. 此外, 全局码本可以看成是局部码本的特殊情况 (可以引入相应的控制参数).

除上述两种码本外, 文献 [3] 认为还有另外两种码本 —— 合成码本和混合码本. 这两种码本对应的编码方法实际上都不是纯分形编码方法, 而是分形编码与 VQ 编码的结合, 属于混合分形编码的范畴.

第四节 亮度变换类型

我们已经知道, 在分形编码中, 编码一幅图像 μ_{org} 就是寻求一个压缩变换 T , 它的不动点 μ_{fix} 是原始图像 μ_{org} 尽可能好的近似图像. 这就是所谓的 IFS 逆问题, 至今没有完全解决. 目前, 克服这个困难的次优方法是借助拼贴定理, 通过图像 μ_{org} 与其拼贴图像 $T\mu_{\text{org}}$ 距离 $d(\mu_{\text{org}}, T\mu_{\text{org}})$ 的极小化, 来近似实现图像 μ_{org} 与不动点图像 μ_{fix} 距离 $d(\mu_{\text{org}}, \mu_{\text{fix}})$ 的极小化.

寻找极小化 $d(\mu_{\text{org}}, T\mu_{\text{org}})$ 的变换 T 的目的, 是为了实现对图像 μ_{org} 的压缩. 因此, 变换 T 除压缩外, 还应该满足其他一些约束条件, 例如:

(1) 变换 T 不能是线性的, 否则压缩性使其不动点只能是零图像 (例如定义于实数轴上的变换 $T: Tx = 0.5x$, 其不动点显然只有 $x = 0$);

(2) 变换 T 应该采用结构简单、计算方便的形式, 一方面使得极小化问题 $\min_T d(\mu_{\text{org}}, T\mu_{\text{org}})$ 易于求解, 另一方面存储它的比特数要大大低于原始图像;

(3) 变换 T 的不动点对于变换参数的量化应该具有鲁棒性 (robust), 因为压缩变换的参数必须量化后存储, 量化后再复原的变换只是原变换的近似变换.

在基本分形编码中, 每个 R 块是用 D 块和常值亮度块的线性组合来编码的: $R \approx s \cdot D + o \cdot 1$. 显然, 按这种方式构造的变换 T 是满足上述条件的最简单变换, 其编码能力是有限的. 为了考虑其推广, 我们用像素值来表达这个向量表示式. 把 R 块 R 和码本块 D 按某种方式 (如行排序) 向量化:

$$R = (r_1, r_2, \dots, r_n)^T, \quad D = (d_1, d_2, \dots, d_n)^T,$$

其中, n 是图像块的像素数. 于是 $R \approx s \cdot D + o \cdot 1$ 按像素表达为

$$r_k = s \cdot d_k + o, \quad 1 \leq k \leq n.$$

这表明, R 块中的像素值是对应的匹配码本块中相应像素值经相同的亮度调整和亮度偏移得到的.

为了得到更好的编码图像质量, 一个自然的推广就是考虑 R 块的像素值 r_k 是匹配码本块像素值的更一般函数 [33]:

$$r_k = t_k(d_1, d_2, \dots, d_n; x_1, x_2, \dots, x_m), \quad 1 \leq k \leq n,$$

其中 x_i 是待定实数, R 块编码中的最小二乘问题现在变成

$$\min_{x_1, \dots, x_m \in \mathbb{R}} \sum_{k=1}^n \{r_k - t_k(d_1, d_2, \dots, d_n; x_1, x_2, \dots, x_m)\}^2. \quad (8.4.1)$$

如果存在一个 $n \times m$ 矩阵 M , 使得

$$\begin{bmatrix} t_1(d_1, d_2, \dots, d_n; x_1, x_2, \dots, x_m) \\ \vdots \\ t_n(d_1, d_2, \dots, d_n; x_1, x_2, \dots, x_m) \end{bmatrix} = Mx, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, \quad (8.4.2)$$

那么, 由此导出的变换 T 仍然满足上述三个条件, 因为 T 具有仿射形式 $T\mu = A\mu + b$, 其中 A 是 $N \times N$ 矩阵, N 是图像 μ 的像素总数.

下面, 我们给出式 (8.4.2) 的几个特殊情况, 它们都已在文献中采用.

(1) 设

$$M = \begin{bmatrix} d_1 & 1 \\ \vdots & \vdots \\ d_n & 1 \end{bmatrix}, \quad x = \begin{bmatrix} s \\ o \end{bmatrix}, \quad (8.4.3)$$

那么, $Mx = [sd_1 + o, \dots, sd_n + o]^T$, 这正是基本分形编码使用的形式, 也是最常用的形式.

(2) 设

$$M = \begin{bmatrix} d_1 - \bar{d} & 1 \\ \vdots & \vdots \\ d_n - \bar{d} & 1 \end{bmatrix}, \quad \bar{d} = \frac{1}{n} \sum_{i=1}^n d_i, \quad x = \begin{bmatrix} s \\ o \end{bmatrix}, \quad (8.4.4)$$

那么, $Mx = [s(d_1 - \bar{d}) + o, \dots, s(d_n - \bar{d}) + o]^T$, 这也是分形图像编码文献中经常使用的改进形式. 因为内积 $(D - \bar{d} \cdot \mathbf{1}, \mathbf{1}) = 0$, 所以 D 的移去均值版本 $D - \bar{d} \cdot \mathbf{1}$ 正交于固定块 $\mathbf{1}$ (亮度均为 1), 其效果是解除了系数 s 和 o 的相关性.

(3) 设

$$M = [D, B_1, \dots, B_p] = \begin{bmatrix} d_1 & b_1^1 & \cdots & b_p^1 \\ \vdots & \vdots & & \vdots \\ d_n & b_1^n & \cdots & b_p^n \end{bmatrix}_{n \times (p+1)}, \quad (8.4.5)$$

其中 B_i 是 p 个与原始图像无关的固定块. 按照这种方式, R 块由 D 和这 p 个固定块的线性组合来编码, 即 $R = x_0 \cdot D + \sum_{i=1}^p x_i \cdot B_i$.

(4) 设

$$M = [D_{i_1}, \dots, D_{i_l}, B_1, \dots, B_p] = \begin{bmatrix} d_1^{i_1} & \dots & d_1^{i_l} & b_1^1 & \dots & b_p^1 \\ \vdots & & \vdots & \vdots & & \vdots \\ d_n^{i_1} & \dots & d_n^{i_l} & b_1^n & \dots & b_p^n \end{bmatrix}_{n \times (l+p)}, \quad (8.4.6)$$

其中 B_i 是与原始图像无关的固定块, D_{i_1}, \dots, D_{i_l} 是码本块. 按照这种方式, R 块由这 l 个码本块和 p 个固定块的线性组合来编码. 为了使计算简便, D_{i_1}, \dots, D_{i_l} 选取为码本的一个正交基, R 块则用其中尽可能少的基块来表示——例如采用匹配追赶算法 (matching pursuit)^[35].

(5) 设

$$M = \begin{bmatrix} d_1^2 & d_1 & 1 \\ \vdots & \vdots & \vdots \\ d_n^2 & d_n & 1 \end{bmatrix}_{n \times 3}, \quad x = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad (8.4.7)$$

其中, d_i^2 表示 d_i 的平方. 显然, R 块的像素值 $r_i = ad_i^2 + bd_i + c$, d_i 是码本块 D 的像素值. 文献 [36] 就采用了这种亮度变换形式.

对于 $r_k = t_k(d_1, d_2, \dots, d_n; x_1, x_2, \dots, x_m)$ 不是 x_i 的线性函数的情形 (不能写成 (8.4.2) 的形式), 目前的研究非常少, 主要原因可能在于极小化问题 (8.4.1) 的求解比较困难. 尽管如此, 仍有学者对此进行了探索^[37].

第五节 变换参数的量化

在分形编码中, 一幅图像是用一个使图像近似不变的压缩变换来表达的, 而变换又是由表达它的参数确定的. 为了进一步实现图像数据的压缩, 表达压缩变换的参数必须进行量化处理, 并对量化参数进行合理的比特分配. 经量化处理后的变换就组成压缩图像的分形码.

我们已经知道, 在现有的分形图像编码中, 编码器是极小化原始图像 μ_{org} 与其拼贴图像 $T\mu_{\text{org}}$ 之间的距离, 而不是与吸引子图像 $\mu_A = T\mu_A$ 之间的距离. 其理论依据是拼贴定理, 但是, 拼贴定理表明前一距离仅仅是后一距离的一个上界 (不计常数因子). 此外, 在极小化距离 $d(T\mu_{\text{org}}, T\mu_{\text{org}})$ 的方法中, 一般采用最小二乘法求尺度系数和偏移系数 (详见第七章), 这种方法是在忽略对尺度系数的约束 (即 $|s| < 1$) 的基础上进行的, 对于不满足约束的尺度系数一般作截断处理. 由此可见, 即使不量化变换参数, 吸引子图像 μ_A 也仅仅是原始图像的一个近似, 或者说不进行量化处理的分形码也是有信息损失的, 是一种有失真码. 现在为了进一步提高压缩比, 必须对变换参数进行量化. 量化是多对一映射, 是不可逆过程, 必然带来量

化误差,从而进一步加大了信息损失.因此,如何量化变换参数以尽量减少量化的负面影响以及进行合理的比特分配也是分形编码不能忽略的问题.

D 块地址和等距变换序号(或虚拟码书的索引号)、适应分割编码中附加的分割信息本身就是用整数表示的,勿需(也不能)进行量化处理,因为它们必须无失真存储或传输.因此,在分形图像编码中,变换参数的量化实际上就是灰度映射中尺度系数和偏移系数的量化问题.

实验表明,尺度系数和偏移系数的分布不是均匀的——实际上,许多标准测试图像对应的尺度系数的值绝大多数分布在 $[-1, 1]$, 其柱状图在 0 的附近形成峰值或多峰值.尽管如此,人们还是对尺度系数和偏移系数采用均匀量化,因为后续熵编码可以补偿这种忽略概率分布的量化的不足.关于尺度系数和偏移系数的优化量化问题,许多学者对此进行了研究.读者可以参看综述文章 [3] 的有关评述,并从中找到相关资料.随便指出,关于尺度系数的优化量化存在一些矛盾的结论.例如,文献 [38] 建议对于较大的尺度系数采用更为精细的量化,这显然与文献 [39] 的 PDF 优化量化器的要求不一致(PDF 是 probability density function 的缩写),因为尺度系数的概率密度函数在 0 的附近呈现中间高、两边低的形状,换言之,大尺度系数出现的概率小于小尺度系数的概率.

对量化尺度系数 s 和量化偏移系数 o 的比特分配方案主要有(单位 bit): 2 和 6(文献 [40]); 5 和 8(文献 [39]); $2 \sim 4$ 和 $3 \sim 8$ (文献 [41]) 以及 5 和 7(文献 [32]). Fisher^[32] 对大量比特分配方案的编码性能进行了比较,结果显示上述最后一种比特分配方案能够提供最好的编码性能.出现这些差异的原因,我们认为可能是比特分配与采用的编码方案(如亮度变换、分割方案和量化方式的选择等)有关,不同的编码方式应该采用不同的比特分配策略.

我们用文献 [32] 描述的基于四叉树分割的分形算法对标准测试图像 256×256 Lena 进行了实验,算法对参数的量化方式采用均匀量化.结果由表 8-1 给出,其中倒数第二、三行中 s 和 o 分别选取 $2 \sim 4$ 和 $3 \sim 8$ 的中位数.算法中的最小深度、最大深度及误差阈值分别取为 4、6 和 2.

从表 8-1 可以看出,对于基于四叉树分割的分形编码算法,综合 PSNR 度量的图像质量和压缩比,尺度系数 s 和量化偏移系数 o 的均匀量化值的比特分配分别为 5 和 7 是最好的.这证实了 Fisher 在文献 [32] 中报告的结果.

表 8-1 不同比特分配时的编码性能

比特分配 (bit allocation)		PSNR	压缩比 (compression ratio)
尺度系数 scale s	偏移系数 offset o		
2	6	28.29	10.43
5	8	29.25	8.77
3	5	28.56	10.30
3	6	28.78	10.12
5	7	29.23	9.13



(1) 原始图像



(2) 解码图像 (s_bit = 2; o_bit = 6)



(3) 解码图像 (s_bit = 5; o_bit = 8)



(4) 解码图像 (s_bit = 3; o_bit = 5)



(5) 解码图像 (s_bit = 3; o_bit = 6)



(6) 解码图像 (s_bit = 5; o_bit = 7)

图 8-5 五种比特分配的主观质量对比

我们已经知道, PSNR 有时并不是可靠的图像质量的完美度量, 为此, 我们给出了上述 5 种比特分配的主观质量对比 (图 8-5)。可以看出, 5 和 7 的比特分配从主观质量上看也是最好的。此外, 量化尺度系数 s 的比特分配越少, “块效应”越明显 (块效应是分形编码、VQ 编码、DCT 编码等分块编码方法的典型特征)。

第六节 分形解码

比较其他的图像编码技术 (如变换编码、矢量量化编码等), 迭代解码是分形编码技术的新颖独特之处。分形解码是一个简单快速的迭代过程, 由 Banach 压缩映射原理给出的迭代算法完成。我们知道, 分形码描述了一个使图像近似不变的压缩仿射变换, 解码器按照分形码恢复出这个压缩仿射变换 T , 然后迭代作用于初始图像 $\mu_0: T^n(\mu_0) = T(T^{n-1}(\mu_0))$, 一般不超过 10 次迭代就可得到收敛图像, 且结果不依赖初始图像, 拼贴定理则保证这幅收敛图像是原始图像的一个近似图像。应该指出, 虽然迭代图像序列的收敛性不依赖初始图像, 但是解码迭代次数与许多因素有关, 如初始图像的选择、原始图像的复杂性等。

分形解码已有坚实的理论基础 (即 Banach 压缩映射原理), 简单快速的迭代解码是分形图像编码技术的显著特征之一, 这也许是分形解码研究不如分形编码活跃的主要原因。诚然, 对于许多应用来说分形解码速度已足够快了, 但是要求超高速解码的应用场合 (如实时视频解码、快速图像恢复等) 则需要更快的解码速度, 可见研究快速解码算法也是某些应用领域的需要。因此, 分形解码的研究自然而然都集中在快速解码算法方面。另一方面, 分形解码是按 Banach 不动点定理给出的迭代方法进行的, 这种迭代解码过程是无法控制的。然而有些应用场合, 例如设计分形编码教学软件、利用计算机辅助技术制作卡通以及窄带渐进传输等, 需要控制解码速度的快慢 (指一次迭代的结果分多次迭代完成)。为此, 作者在文献 [42] 中首次引入“渐进 / 可控分形解码”的概念, 并提出了一个渐进分形解码算法, 实现了对分形解码过程的部分控制。

一、分级解码

分形码描述的压缩仿射变换在不同分辨率下的不动点图像之间有密切的联系。为此, Barahav 等 [43] 给出了分形码的一种分级解释 (hierarchical interpretation), 其理论基础是该文作者证明的一个定理: 对于一个给定的分形码, 存在唯一的有界连续值函数 $f(x) \in L^\infty[0, 1]$, 称为 IFS 嵌入函数 (IFS embedded function), 使得一个 N 维向量 V_N 是该分形码的不动点的充要条件是

$$V_N(j) = f_N(j), \quad j = 1, 2, \dots, N, \quad (8.6.1)$$

其中, $X_N(j)$ 表示 N 维向量 X_N 的第 j 个分量, 而 $f_N(j)$ 是连续值函数 $f(x)$ 在区间 $[(j-1)/N, j/N]$ 上的平均值, 即

$$f_N(j) = N \int_{(j-1)/N}^{j/N} f(x) dx. \quad (8.6.2)$$

上式可以看成是对连续值函数 (模拟信号) 的一种广义采样, 表示连续值函数 $f(x)$ 在分辨率 N 下的逼近. 根据这个定理, 容易证明下面的推论: 设向量 V_N 是一个分形码在 N 维空间 R^N 中的不动点, 那么 $V_{N/2}$ 是同一个分形码在 $N/2$ 维空间 $R^{N/2}$ 中的不动点, 其中

$$V_{N/2}(j) = \frac{1}{2} (V_N(2j-1) + V_N(2j)), \quad j = 1, 2, \dots, N/2. \quad (8.6.3)$$

这给出了从高分辨向量计算低分辨向量的计算公式 (zoom-out 算法). 此外, Barahav 等还建立了如何从低分辨向量 ($V_{N/2}$) 得到高分辨向量 (V_N), 也给出了一个确定的计算公式 (zoom-in 算法). 这些结论表明, 同一个分形码在不同维数空间 R^n 中的不动点 V_N 与 $V_{N/2}$ 之间存在确定的相互联系.

根据分形码的上述分级解释, Barahav 等提出了一个快速解码方案, 解码迭代可以先在图像的低分辨版本上进行, 得到低分辨不动点图像后, 按照确定算法即可生成高分辨的原始图像. 因为解码迭代是对低维向量进行的, 因此大大节约了解码时间. 顺便指出, 容易看出这种解码方式对编码有一定的限制, 也就是生成 D 块池的步长必须等于 R 块的边长.

二、Gauss-Seidel 型解码

我们已经知道, 分形编码器给出的分形码描述了一个使图像近似不变的压缩仿射变换, 用矩阵形式写出来就是

$$T: R^N \rightarrow R^N, \quad T(x) = Ax + b, \quad (8.6.4)$$

其中 A 是一个 $N \times N$ 矩阵, $x \in R^N$ 是原始图像, $b \in R^N$ 是 N 维实值向量. 解码器的工作就是要求出上述仿射变换的不动点 $x_f = T(x_f)$, 即求解下面的矩阵方程:

$$x = T(x) = Ax + b. \quad (8.6.5)$$

通常的方法是采用 Banach 不动点定理给出的迭代算法进行: 设 $x^{(k)} = T^k(x_0)$ 表示变换 T 的 k 次迭代结果, $x_0 \in R^N$ 是任意初始向量, 则变换 T 的 $k+1$ 次迭代结果由下面的方程计算:

$$x^{(k+1)} = Ax^{(k)} + b. \quad (8.6.6)$$

这个序列的极限就是变换的不动点: $x_f = \lim_{k \rightarrow \infty} x^{(k)}$. 应该指出, 矩阵 A 通常是一个高阶数的稀疏矩阵 (如 256×256 、 512×512 等), 在分形解码器中并不储存 (也不需要) 这个矩阵. 这里引进这个矩阵仅仅是想说明解码的数学原理而已.

因为 A 是一个稀疏矩阵, 矩阵方程 (线性方程组) $x = Ax + b$ 的求解还有更好的算法, 例如 Gauss-Seidel 迭代法. 下面回顾这种迭代算法, 为方便计, 假设矩阵方程已变换为如下的形式 (主对角元全为 1):

$$\begin{pmatrix} 1 & a_{12} & \cdots & a_{1N} \\ a_{21} & 1 & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}, \quad (8.6.7)$$

它可以表示成如下的形式:

$$(I - L - U)x = b, \quad (8.6.8)$$

其中, I 是 $N \times N$ 单位矩阵, L, U 分别是下三角矩阵和上三角矩阵.

所谓 Gauss-Seidel 迭代法, 就是指矩阵方程 (8.6.8) 的解可以由下面的迭代近似计算:

$$x^{(k+1)} = b + Lx^{(k+1)} + Ux^{(k)}. \quad (8.6.9)$$

以分量形式写出上面的方程, 例如它的第 i 个方程:

$$x_i^{(k+1)} = b_i - a_{i1}x_1^{(k+1)} - \cdots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{iN}x_N^{(k)}. \quad (8.6.10)$$

由此不难看出, 在确定 $x_i^{(k+1)}$ 时利用了前 $i-1$ 个分量的修正值 $x_j^{(k+1)}$ ($j \leq i-1$). 可见, Gauss-Seidel 迭代法是一种分步修正算法 (successive correction).

根据 Gauss-Seidel 迭代法, 文献 [44, 45] 提出了新的解码算法. 这种解码算法是 Gauss-Seidel 迭代法的直接应用, 因此, 我们把它们统称为 Gauss-Seidel 型解码. 文献 [44] 的作者在技术报告 [46] 中指出, 与基于迭代 (8.15) 的传统解码相比, Gauss-Seidel 型解码有两个优点: 一是储存量减少一半; 二是只要所有的对比度因子具有相同的正负号, 新的解码有更高的收敛速率. 此外, 对大量测试图像的实验结果表明, 即使并非所有的对比度因子都具有相同的正负号, 解码速度也都快于传统解码算法.

三、基于初始图像选择的快速解码

分形解码是相对简单的迭代过程, Banach 不动点定理保证始于任意初始图像的迭代序列都收敛于 (唯一) 不动点图像, 换言之, 极限图像 (不动点图像) 与初始

图像的选择无关. 但是, 迭代序列的收敛速度极大地依赖于选择的初始图像 (我们曾在上一章指出过这个事实).

基于上述事实, 文献 [47] 提出了一个基于初始图像选择的快速解码算法. 该文通过实验证实, 解码收敛的迭代数目可以通过选择一个适当的初始图像来减少, 而且初始图像为原始图像对应的块均值图像 (R-averaged Image) 时导致了最快的解码速度. 其中, 块均值图像是所有 R 块都是常值块的图像, 具体说来, 设图像由一组不重叠块 R_i 组成:

$$\mu = \bigcup_i R_i, \quad R_i \cap R_j = \emptyset \quad (i \neq j). \quad (8.6.11)$$

如果 $R_i = \bar{R}_i \cdot \mathbf{1}_i$, 那么图像 μ 称为块均值图像. 这里 \bar{R}_i 是图像块 R_i 的亮度均值, $\mathbf{1}_i$ 是与 R_i 同型的全一图像块 (亮度都是 1).

我们知道, 传统分形解码按下面的方式进行:

$$R_i^{(k)} = s_i \cdot D_{m(i)}^{(k-1)} + o_i \cdot \mathbf{1}, \quad \mu^{(k)} = \bigcup_i R_i^{(k)}, \quad (8.6.12)$$

其中, $R_i^{(k)}$ 是第 k 次迭代图像 $\mu^{(k)}$ 的 R 块, $D_{m(i)}^{(k-1)}$ 是源于第 $(k-1)$ 次迭代图像 $\mu^{(k-1)}$ 的码书块 ($m(i)$ 指明 $D_{m(i)}^{(k-1)}$ 是 $R_i^{(k)}$ 的最佳匹配块). 文献 [47] 把传统解码 (8.21) 分解为 DC 解码和 AC 解码两部分:

$$\begin{aligned} R_i^{(k)} &= s_i \cdot \left(\bar{D}_{m(i)}^{(k-1)} \cdot \mathbf{1} + \bar{D}_{m(i)}^{(k-1)} \right) + o_i \cdot \mathbf{1} \\ &= \left(s_i \cdot \bar{D}_{m(i)}^{(k-1)} + o_i \right) \cdot \mathbf{1} + s_i \cdot \bar{D}_{m(i)}^{(k-1)}, \end{aligned} \quad (8.6.13)$$

其中, “—” 和 “~” 分别表示 DC 分量和 AC 分量. 按照式 (8.6.13), DC 解码过程和 AC 解码过程被分别定义如下:

$$\bar{R}_i^{(k)} = \left(s_i \cdot \bar{D}_{m(i)}^{(k-1)} + o_i \right) \cdot \mathbf{1}, \quad \tilde{R}_i^{(k)} = s_i \cdot \bar{D}_{m(i)}^{(k-1)}. \quad (8.6.14)$$

接着该文证明 DC 解码的吸引子是原始图像的块均值图像的近似图像 (大约需 5 次迭代). 同时, 实验表明, 一次 DC 解码迭代的计算量约为一次传统迭代的 15%, 因此, 5 次 DC 解码迭代的计算量约为一次传统迭代的计算量.

基于上述实验与理论分析, 文献 [47] 的快速解码过程分成两个阶段进行. 第一阶段进行 DC 解码, 近似得到原始图像的块均值图像; 第二阶段按传统解码进行, 初始图像取为第一解码阶段得到的原始图像的近似块均值图像. 对标准测试图像——Lena 和 Boat 图像的实验结果表明, 第二阶段解码在第三次迭代收敛. 同时考虑两个解码阶段, 新的解码方案仅仅需要大约 4 次迭代即可收敛, 这大大低于传统解码的 10 次迭代.

我们对上述快速解码算法进行了改进. 与原算法不同之处在于, 改进算法在第一阶段得到原始图像准确的 D 块均值图像, 而不仅仅是一个近似. 因此, 改进算法实现了理想收敛 (ideal convergence)^①. 下面介绍这个改进算法.

给定一个 R 块 $R \in R^n$ 和一个码块 $D \in \Omega \subset R^n$, 设 $MSE(R, D)$ 是 R 与 D 的亮度变换版本 $a \cdot D + b \cdot 1$ 之间的最小均方误差 (mean square error, MSE):

$$MSE(R, D) = \frac{1}{n} \min_{a, b \in \mathbb{R}} \|R - (a \cdot D + b \cdot 1)\|^2 = \frac{1}{n} \|R - s \cdot D - o \cdot 1\|^2. \quad (8.6.15)$$

显然, 式 (8.6.15) 的内层极小化问题实际上是二元多项式的最小值问题. 于是, 下面的等式能够通过直接的计算来推出:

$$o = \bar{R} - s \cdot \bar{D}, s = \langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle / \|D - \bar{D} \cdot 1\|^2, \quad (8.6.16)$$

其中, $\langle \cdot, \cdot \rangle$ 表示向量内积, \bar{X} 是子块 X 的亮度均值.

为了在解码端准确得到原始图像的 R 块均值图像, 我们必须解偶 (8.6.16) 中的两个参数 s 和 o . 为此, 把 $o = \bar{R} - s \cdot \bar{D}$ 代入式 (8.6.15), 得到

$$MSE(R, D) = \frac{1}{n} \|R - s \cdot (D - \bar{D} \cdot 1) - \bar{R} \cdot 1\|^2. \quad (8.6.17)$$

于是, 传统分形解码过程 (8.6.12) 被修改为

$$R_i^{(k)} = s_i \cdot (D_{m(i)} - \bar{D}_{m(i)} \cdot 1)^{(k-1)} + o'_i \cdot 1, o'_i = \bar{R}_i, \quad (8.6.18)$$

这个修改解码过程将在第二阶段用于重构原始图像. 实际上, 这种去均值解码过程也是分形编码文献常常使用的解码形式^[3]. 注意, R 块的分形码现在是 $(m(i), \bar{s}_i, o'_i)$ 而非 $(m(i), \bar{s}_i, \bar{o}_i)$.

一个自然的问题: 重构原始图像是解码器的工作, 解码器自然不能访问原始图像, 那如何准确得到原始图像的 R 块均值图像呢? 注意到 $o'_i = \bar{R}_i$ 是 R 块的亮度均值 (由分形码提供), 因此原始图像的 R 块均值图像可以准确得到

$$\bar{\mu} = \cup_i R_i : R_i = o'_i \cdot 1. \quad (8.6.19)$$

显然, 与式 (8.6.18) 表示的解码过程相比, “拼贴” $\bar{\mu}$ 的时间几乎可以忽略.

当初始图像为原始图像的 R 块均值图像时, 传统解码过程两次迭代就可得到收敛图像 (理想收敛)^[47]. 但是, 因为改进算法采用去均值解码过程 (8.6.18), 因此, 我们必须实验验证改进算法能够实现理想收敛.

为简单起见, 基于固定方块分割的基本分形算法被用于编码 256×256 Lena 和 Peppers 图像, 以得到修改后的分形码 $\cup_i (m(i), \bar{s}_i, o'_i)$, 其中, R 块的大小取为 4×4 , 纵横方向步长取为 8 个像素点. 重构图像质量由 PSNR 度量.

^① 在文献 [47] 中, 初始图像为原始图像的 R 块均值图像时的收敛称为“理想收敛”, 一般两次迭代即可得到收敛图像.

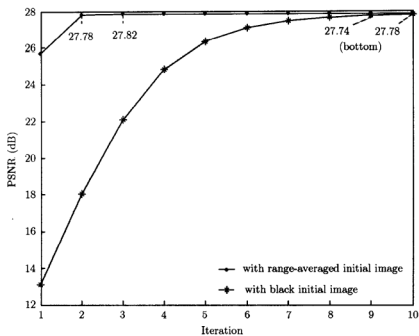
图 8-6 给出了第一阶段解码的输出图像, 它们是原始图像的 R 块均值版本. 当初始图像为原始 Lena 图像的 R 块均值版本时, 第 2 次与第 10 次解码图像显示在图 8-7 中, 它清楚地显示这两幅图像几乎没有差别. 图 8-8 给出了收敛过程的量化表示, 其中, “Iteration” 表示第二阶段解码的迭代次数, “PSNR” 表示原始图像与始于原始图像 R 块均值版本或全黑图像的各次迭代图像之间的峰值信噪比. 图 8-8 清楚地显示, 当初始图像为原始图像的 R 块均值版本时, 在 0.05dB 的精度范围内, 收敛图像在第 2 次迭代得到.



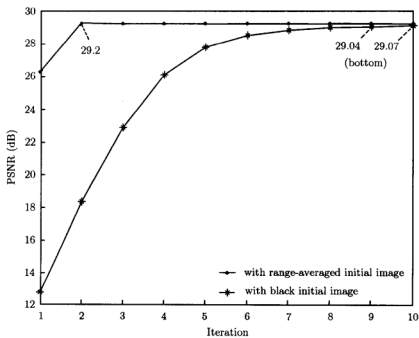
图 8-6 第 1 阶段生成的原始图像的 R 块均值图像: (左) Lena 图像; (右) Peppers 图像



图 8-7 以原始 Lena 图像的 R 块均值版本为初始图像时的解码图像: (a) 第 2 次迭代;
(b) 第 10 次迭代



(a) Lena



(b) Peppers

图 8-8 收敛图: (a) Lena 图像; (b) Peppers 图像

我们对基于初始图像选择的改进快速解码算法作个总结。在改进算法中, 原始图像的 R 块均值版本在解码器中被准确得到, 而在原算法中仅仅得到 R 块均值版本的一个近似。实验表明, 改进算法实现了文献 [47] 提出的“理想收敛”。此外, 改进算法与原算法比较有两个优点: 一是不需要复杂的理论分析; 二是解码过程就是传统解码过程的微小修改 (解码器结构不变), 因此, 它能够以直接的方式融入其他的分形编码算法。

应该指出, 除上述解码方案以外, 还有其他一些快速解码方法 (如不迭代解码等), 请参考文献 [47] 的引言部分。

四、渐进/控制解码

对于窄带传输 (low bandwidth transmission), 渐进解码 (progressive decompression) 是十分有用的, 拥有分形图像编码技术多项美国专利的 Barnsley 迭代系统公司 (Iterated Systems, Inc.) 对于针对窄带传输的渐进解码曾做过许多工作 (商用)。然而, 传统分形解码并不直接适用于这种情况, 因为解码依据的 Banach 不动点定理没有为不动点迭代过程提供任何控制参数, 换言之, 传统分形解码不能以适当的方式被控制。为了克服这个困难, 这家公司的渐进解码方案需要附带一些由一个特殊的编码器以适合渐进解码的方式编码的文件, 同时需要一个特殊的解码器来读取这些文件。此外, 控制 / 渐进分形解码对某些多媒体应用也是有用的, 例如, 在利用计算机辅助技术制造卡通中, 一幅图像有时需要从另一幅图像出发以渐进的方式被逐渐显示出来, 这需要控制 (放慢) 解码过程。传统分形解码也不直接适用于这种情况, 因为上述同样的原因。因此, 考虑其他可能的解码方案是有意义的, 例如, 它们能够以不同的方式显示迭代重构过程, 或者按人们的需要控制解码过程, 或者以渐进方式展示每次迭代新增的图像细节, 等等。

为了实现上述目标, 我们在文献 [42] 中提出了一个新颖的渐进 / 控制分形解码方案, 它基于一个新的渐进 / 控制不动点定理 (定理 6.2.6)。实验结果显示, 新方案能够实现上面提到的部分目标。

在传统分形编码中, 解码方案总是使用源于 Banach 不动点定理的迭代过程:

$$\mu_{\text{org}} \approx \bar{\mu} = \lim_{n \rightarrow \infty} T^n(\nu) \approx T^N(\nu), \forall \nu \in X. \quad (8.6.20)$$

基于定理 6.2.6 的新不动点迭代过程, 我们的解码方案为

$$\mu_{\text{org}} \approx \bar{\mu} = \lim_{n \rightarrow \infty} \mu_n \approx \mu_N, \quad (8.6.21)$$

其中,

$$\mu_0 = \nu, \mu_n = (1 - \lambda)\mu_{n-1} + \lambda T(\mu_{n-1}), \lambda \in (0, 1], \quad \forall \nu \in X. \quad (8.6.22)$$

在式 (8.6.20) 和 (8.6.21) 中, $\nu \in X$ 可以任意选择, 例如一幅全黑图像, N 通常选择足够大以满足所需的精度。

注意, 式 (8.6.21) 是式 (8.6.20) 的特殊情况 ($\lambda=1$)。此外, 新的解码方案并不需修改现有的分形编码过程, 因此可以适合于任何分形编码方法。正如实验研究所示, 控制参数 λ 在新的解码方案中是一个重要的因子, 它引入了解码过程的灵活性和可控性。因此, 新方案有希望成为分形图像压缩技术的一个改进的解码方案。

图 8-9(a) 和图 8-9(b) 分别给出了传统解码与新解码方案的框图。

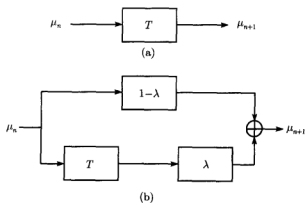


图 8-9 解码框图: (a) 现有方案; (b) 新方案

实现方案选择基于适应四叉树算法, 我们使用 Fisher 的四叉树编码器^[32]进行所有的实验, 因为这种四叉树编码器为本文的实验目的提供了好的测试平台。由于基于定理 6.2.6 的解码方案并不修改分形编码过程, 因此, 我们只需修改 Fisher 四叉树编码器的解码部分。尽管取不同控制参数的新解码算法测试了大量不同的图像, 但是这里仅仅给出, 当参数分别取 0.25、0.50、0.75 和 1.00 时, 新解码算法测试标准图像 256×256 Lena 的实验结果。

我们研究了不同控制参数对新解码过程的影响, 结果显示在图 8-10 中。图中“iteration”表示解码迭代次数, “PSNR”表示原始 Lena 图像与始于全黑图像的各次迭代图像之间的峰值信噪比。当控制参数为 1 时, 在 PSNR 为 0.01dB 的精度范围内, 解码过程在第 7 迭代已收敛 (我们的实验表明, 第 7 次迭代图像的 PSNR 为 32.95dB, 以后保持为 32.96dB)。因此, 如果定义 0.01dB 的收敛阈值, 始于全黑图像的解码迭代在第 7 次就得到高质量的重构图像。此外, 对于始于其他初始图像、并获得高质量重构图像的解码迭代次数也大致相同。这显示传统解码的确是相当快的。然而当控制参数 $\lambda < 1$ 时, 由 PSNR 度量的解码收敛被减慢; 控制参数 λ 越小, 收敛越慢 (见图 8-10)。特别地, 当控制参数 $\lambda = 0.25$ 时, 随着迭代次数的增加 PSNR 几乎是线性增加。

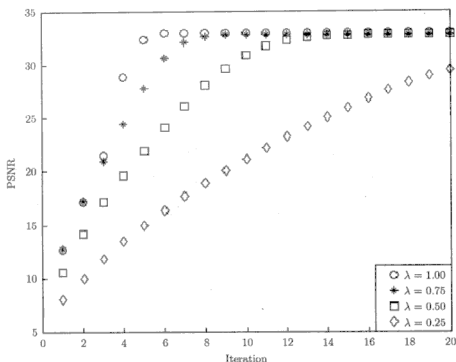


图 8-10 不同参数时的解码器速度

为了显示图像细节是如何随着迭代的增加被逐步加入的, 我们把相邻解码图像的差作为图像来显示. 图 8-11 给出了控制参数为 0.50 时相邻解码图像的差图像 $\mu_{n+1} - \mu_n$ ($0 \leq n \leq 10$), 其中, 从左到右、自上而下, 迭代次数 n 分别是 0, 1, \dots , 10; 此外, 为了清楚显示, 每幅差图像都加了亮度值 100.

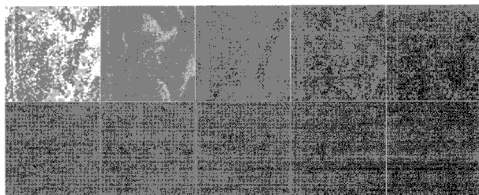
图 8-11 解码图像的差图像 $\mu_{n+1} - \mu_n$ 的可视化 ($\lambda = 0.50$)

图 8-12~图 8-15 分别显示了不同控制参数值时,一幅高质量的图像是如何从一幅全黑图像得到的。

图 8-12 给出了控制参数为 0.25 时的解码图像序列,其中,从左到右、自上而下,第 1 幅是初始图像,接着依次为第 1 至 17 次解码图像,然后是第 25 次解码图像,最后一幅是原始图像。它显示解码很慢,但清楚地显示 Lena 图像是如何以渐进的方式从全黑图像演变而来。

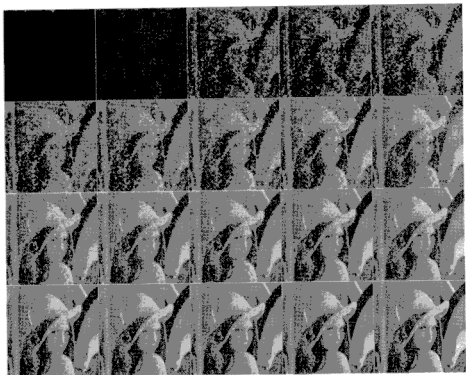
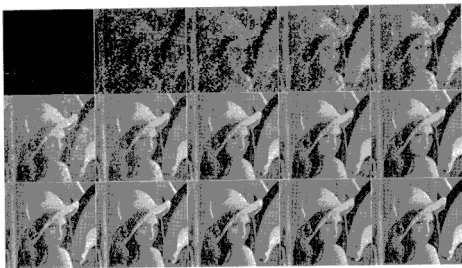
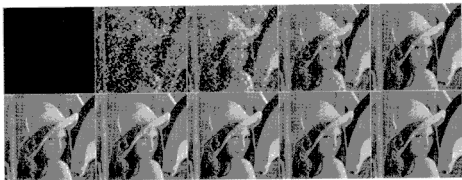
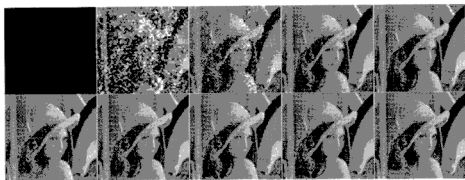


图 8-12 $\lambda = 0.25$ 时的解码图像序列

图 8-13 给出了控制参数为 0.50 时的解码图像序列,其中,从左到右、自上而下,第 1 幅是初始图像,接着依次为第 1 至 12 次解码图像,然后是第 20 次解码图像,最后一幅是原始图像。类似与控制参数为 0.50 时的图 8-12,它能够显示 Lena 图像是如何从全黑图像渐进演变而来。

控制参数为 0.75 和 1.00 时的结果分别显示在图 8-14 和图 8-15 中。两种情形都不能清楚地显示一幅高质量的图像是如何从全黑图像渐进演变而来,因为解码太快。在图 8-14 中,从左到右、自上而下,第 1 幅是初始图像,接着依次为第 1 至 7 次解码图像,然后是第 20 次解码图像,最后一幅是原始图像。在图 8-15 中,从左到右、自上而下,第 1 幅是初始图像,接着依次为第 1 至 7 次解码图像,然后是第 10 次解码图像,最后一幅是原始图像。

图 8-13 $\lambda = 0.50$ 时的解码图像序列图 8-14 $\lambda = 0.75$ 时的解码图像序列图 8-15 $\lambda = 1.00$ 时的解码图像序列

我们提出的渐进分形解码方法基于一个新的不动点迭代定理(定理 6.2.6)且现有的分形解码方案是新方案的特殊情况. 实验结果显示, 新方案能够实现可控分形解码, 为需要渐进解码的应用场合(如窄带传输、制造卡通等)提供了一个好的选择. 新解码方案适应于各种分形编码算法, 因为它无需修改现有的分形编码过程.

对于渐进分形解码, 还有许多有趣的问题需要进一步研究. 例如, 一个方向是研究初始点与不动点之间的其他插值方式, 如用单调增加的参数序列 $\{\lambda_n\} (0 < \lambda_n \leq 1)$ 以代替常数参数 λ , 定理 6.2.6 是否仍然成立? 这对进一步改进新的解码方法是有用的, 因为解码图像细节的主要在开始几步被加入(见图 8-11). 此外, 应该指出, 我们的研究是可控/渐进分形解码的初步理论研究, 可能的工程应用需要在将来的研究中探索.

第七节 最优分形编码

数据压缩编码本质上是一个优化问题. 其目标是, 在满足某些质量约束的条件下, 寻找数据的尽可能简短的描述——即在质量容许的条件下尽可能减少描述数据的比特数; 或者在描述数据的比特数的约束下, 寻找质量尽可能好的数据表达方式; 或者更为理想的是, 寻找数据的既尽可能简短的、质量又尽可能好的数据表达方式. 通常, 人们把数据的压缩表达方式限制在某个特殊的形式, 如矢量量化编码、变换编码以及分形编码等. 同样的道理, 从数据表达方式看, 尽管图像的分形编码就是用压缩变换来表达图像数据, 然而, 寻求这样的压缩变换已证明是非常困难的问题. 此外, 为了实现图像数据的压缩以及计算可行, 人们不得不把压缩变换限制为尽量简单的类型. 最简单的变换莫过于线性变换, 其次为仿射变换. 然而变换是不能为线性的, 因为压缩性使线性变换的不动点图像只能是零图像, 因此, 分形编码把变换类型限制为压缩仿射变换. 尽管如此, 问题依然没有解决, 因为仿射变换的不动点图像虽然可以从 Banach 不动点定理给出的迭代算法快速迭代生成, 但它通常不能由变换参数简单地表示出来. 因此, 我们仍然不能直接通过极小化变换参数使得不动点图像尽可能接近原始图像. 尽管拼贴定理为克服这个困难提供了一个可能的解决途径, 但拼贴编码方案也远不是最优的, 因为拼贴定理仅仅表明 $d(\mu_{\text{org}}, \mu_T)$ 不超过 $(1-s)^{-1}d(\mu_{\text{org}}, T\mu_{\text{org}})$ (其中 μ_T 是压缩变换 T 的不动点, s 是变换 T 的压缩因子), 极小化拼贴误差并不意味着极小化吸引子误差. 实际上, 因为迭代解码时的误差传播, 拼贴误差通常小于吸引子误差. 图 8-16 给出了由这两个误差计算的 PSNR 对比图(有关数据源于文献 [33]), 从图中可以看出, 在相同的压缩比下, 按拼贴误差计算的 PSNR 都大于按吸引子误差计算的 PSNR. 换言之, 在相同的压缩比下, 拼贴误差都小于吸引子误差. 此外, 目前还没有理论给出 $d(\mu_{\text{org}}, \mu_T)$ 和 $d(\mu_{\text{org}}, T\mu_{\text{org}})$ 之间更进一步的关系. 这些事实充分表明, “最优”分形编码是一个远未解决的复杂问题. 因此, 我们限于讨论基于拼贴定理的分形编码的最优编码问题.

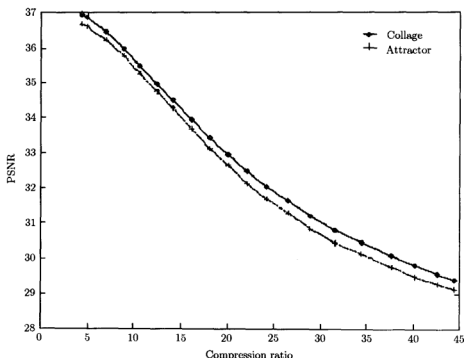


图 8-16 四叉树分形编码器的率失真曲线图, 测试图像是 512×512 Lena

我们先回顾一下基于拼贴定理的分形编码. 以基本分形算法为例, 首先将图像分割成一系列互不重叠且覆盖整幅图像的 R 块, 并从同一图像中选取大小为 R 块两倍边长的 D 块以构成 D 块池. 对于每个 D 块, 采用四邻域像素值平均缩小 D 块到 R 块的尺寸, 然后对收缩后的 D 块进行 8 种等距变换 (4 个旋转与反射), 这样每个 D 块产生 8 个子块. 换句话说, 这样的子块由下面的块变换确定:

$$t_i(R_i) = s_i \cdot (\tau_k \circ S)(D_j) + o_i \cdot 1, \quad (8.7.1)$$

其中, S 是四邻域像素值平均算子, τ_k 是 8 个等距变换之一. $s \cdot X + o \cdot 1$ 称为亮度变换, s 和 o 分别称为对比度因子和亮度偏移. 此外, 为了保证变换是压缩的, 一般要求对比度因子满足约束 (仅仅是充分条件): $|s| \leq s_{\max}$, $0 < s_{\max} < 1$.

对于每个 R 块 R_i , 在上述子块集合中搜索与之最相似的子块 $t_i(R_i)$, 然后由它近似代替 R_i . 这要求解下面的最优化问题:

$$(s_i, o_i, k_i, j_i) = \arg \min_{j,k} \min_{s,o} \|R_i - (s \cdot \tau_k \circ S(D_j) + o \cdot 1)\|_2^2, \quad (8.7.2)$$

这里 j_i 是 R_i 的最佳匹配块的地址, k_i 是最佳等距变换的编号.

满足问题 (8.7.2) 的四元组 (s_i, o_i, k_i, j_i) 的全体就描述了表达原始图像的压缩仿射变换, 它们构成原始图像的分形码. 不动点图像由这个压缩变换迭代作用于任意初始图像来生成, 拼贴定理保证这个不动点图像是原始图像的近似图像 (但近似的程度是无法控制的).

然而, 不动点图像 μ_T 与原始图像 μ_{org} 的误差 (以 MSE 度量, 以下称为吸引子误差) 应该为 (设图像尺寸为 $N \times N$)

$$MSE(\mu_{\text{org}}, \mu_T) = \frac{1}{N^2} \sum_{i,j=1}^N [(\mu_T)_{i,j} - (\mu_{\text{org}})_{i,j}]^2, \quad (8.7.3)$$

而不是拼贴误差

$$MSE(\mu_{\text{org}}, T\mu_{\text{org}}) = \frac{1}{N_R} \sum_{i=1}^{N_R} \|R_i - (s_i \cdot D_{m_i} + o_i \cdot 1)\|_2^2, \quad N_R \text{ 是 } R \text{ 块的个数}. \quad (8.7.4)$$

因此, 许多学者把上述方案称为拼贴编码, 因为最优化问题 (8.7.2) 实际上对应于极小化拼贴误差 $MSE(\mu_{\text{org}}, T\mu_{\text{org}})$. 拼贴编码仅仅是最优分形编码的一个次优解法.

最优分形编码 (或称吸引子编码) 的正式提法. 待编码图像 μ_{org} 的可行分形码 σ 的集合是

$$\Pi = \{(s_1, o_1, k_1, j_1), \dots, (s_{N_R}, o_{N_R}, k_{N_R}, j_{N_R}) | s_i \in [-s_{\max}, s_{\max}], \\ o_i \in R, k_i \in [1, 8], j_i \in [1, N_D]\}, \quad (8.7.5)$$

其中, N_R 、 N_D 分别是 R 块、 D 块的个数. 图像 μ_{org} 的最优分形码 σ_{opt} 就是下面最优化问题的解

$$\mu_{\sigma_{\text{opt}}} = \arg \min_{\sigma \in \Pi} MSE(\mu_{\text{org}}, \mu_{\sigma}), \quad (8.7.6)$$

其中, μ_{σ} 是可行分形码 σ 描述的压缩变换的不动点图像.

文献 [48] 证明, 上述最优分形编码是 NP- 难的. 此外, 拼贴编码也不是最优分形编码的逼近算法, 即对于任意 $\rho > 0$, 存在图像 μ_{org} 和分形码 $\sigma \in \Pi$, 使得

$$MSE(\mu_{\text{org}}, \mu_T) > \rho \cdot MSE(\mu_{\text{org}}, \mu_{\sigma}), \quad (8.7.7)$$

其中, μ_T 是拼贴编码方案得到的压缩变换 T 的不动点图像, μ_{σ} 是最优分形编码确定的分形码 σ 描述的压缩变换的不动点图像. 换言之, 拼贴编码误差与吸引子编码误差之比可能为任意大. 这从理论上证明了拼贴编码不但不会是最优的, 而且对某些图像的编码在理论上还是“不可靠”的. 但是文献 [49] 证明, 拼贴编码在一定的条件下是最优的.

应该指出, 我们在这里提及文献 [48] 的结果, 并不是想借此否定拼贴编码, 而仅仅是想说明目前的分形编码还有许多理论问题需要解决. 事实上, 尽管拼贴编码理论上不完善, 但它有相对简单的优点, 而且在对自然图像的压缩编码中取得了很好的结果, 因此, 拼贴编码仍然是绝大多数分形编码方法的首选, 成为分形编码事实上的标准算法.

第八节 快速分形编码

在分形编码中, D 块池由源于原图像、尺寸通常大于 R 块的图像块组成. 通过应用 8 种等距变换 (旋转与对称反射) 于每个 D 块, D 池扩大为原来的 8 倍. 最后像素平均或欠采样运算 (并收缩图像支持) 使这些图像块的尺寸与 R 块的尺寸完全相同, 这样得到的图像块的全体称为码书 (记为 Ω).

编码的一个重要步骤就是对每个 R 块在 Ω 中搜索最好匹配块 (在某种度量意义下, 通常是 MSE (Mean-Square Error)). 因此, 基本的分形编码与 VQ 编码十分类似. 在 VQ 编码中, 在编码器和解码器都有一本相同的码书, 与此不同, 分形编码的解码器并不存在这样的码书, 而是以自参考方式 (Self-referential Manner) 暗含于图像中. 故有的文献把上述的码书称为 “虚拟码书” (Virtual Codebook). 可以看出, 分形编码的编码时间复杂度非常高, 编码过程十分耗时, 因为对于每个 R 块都要在码书 Ω 中搜索匹配的定义域块, 而理论上可以作为候选的定义域块的数目一般是很大的. 例如, 对最简单的方块分割来说, $N \times N$ 图像的任意尺寸方块数目的阶是 $O(N^3)$. 此外, 对于原始图像分割中的每个 R 块, 如果采用全搜索 (Exhaustive Search, 搜索 Ω 中的所有子块), 搜索花费的时间是 $O(|\Omega|)$ (线性依赖于 $|\Omega|$), $|\Omega|$ 是 Ω 中图像块的数目. 若能减少 Ω 中图像块的数目, 例如对可容许定义域块做出某些限制 (如对固定方块分割来说, 限制定义域块的尺寸和位置等), 或者按灰度变化情况对 Ω 进行某种分类 (R 块的匹配仅仅在 Ω 的某子类中进行), 或者赋予 Ω 合适的结构等, 就有可能降低编码的复杂性, 从而加快编码速度.

目前已提出了多种降低编码时间复杂性 (加快编码速度) 的方法, 我们简要介绍其中几个有代表性的方法. 为了讨论方便, 我们假设图像按固定块分割划分为尺寸 $N_R \times N_R$ 的 R 块, 其他分割方式可类似讨论 (基本原理是一样的).

一、限制 D 块尺寸和位置等的方案

如前所述, 分形编码的一个重要步骤就是对每个 R 块都要在码书 Ω 中搜索最好匹配块, 如果不对定义码书 Ω 的可容许 D 块做出某些限制, 那么可以作为候选的 D 块的数目将是十分大的, 这势必导致编码过程的时间消耗. 一种限制可容许 D 块的简单方法是限制 D 块的尺寸和位置等.

对 $B \times B$ 固定方块分割来说, 如何选取可容许 D 块的尺寸? 这是一个值得详细研究的问题. 我们知道, 表达图像分形码的变换的压缩性并不要求 D 块的几何收缩性, 因此, 可容许 D 块的尺寸理论上可以任意选择. 在多数分形编码文献中, D 块的尺寸通常选为 R 块尺寸 (边长) 的两倍, 即 $2B \times 2B$, 也有文献把 D 块选为与 R 块同大小 (即 $B \times B$)^[50].

如果 D 块的尺寸选为 R 块尺寸 (边长) 的两倍, D 块的位置可以由 D 块之间的间隔 δ 决定. δ 可以小到一像素 ($\delta=1$), 也可以大到 $2B$ ($\delta=2B$). 选取 $\delta=1$ 的方案将产生十分庞大的 D 块池, 这会导致后来的搜索非常耗时. 在实际应用中一般选取 $\delta=B$ 或 $2B$, 尽管它们都可以改进收敛性能, 但是在保真度 (fidelity) 和压缩比方面, 前者 $\delta=B$ 优于后者 $\delta=2B$ ^[51]. 因此, 在分形编码的 VLSI 结构研究中 $\delta=B$ 被优先选用^[52].

在分形图像编码实现中, D 块池通常应用 8 种等距变换扩大为原来的 8 倍. 这样做毫无疑问会改进率失真曲线, 因为在扩大后的码书中有更大的可能性找到匹配更好的码书块, 但编码时间的开销也会同步扩大. 文献 [53] 对此问题进行了系统的研究, 结论是引入 8 种等距变换陡增了编码复杂性, 相同或更好的编码质量可以通过减小相邻 D 块的间隔 δ (从而也扩大了码书) 来实现, 无需 8 种等距变换.

仅仅作上述处理远远不够, 因为这样得到的码书仍然很大. 设图像尺寸为 $N \times N$, 采用固定方块分割, R 块尺寸为 $B \times B$, 可容许 D 块的尺寸选为 $2B \times 2B$, D 块之间的间隔为 δ , 不难计算出 D 块的个数为 $N_d = [(N - 2B)/\delta + 1]^2$, 码书大小 $|\Omega| = 8[(N - 2B)/\delta + 1]^2$. 例如, 设 $B=4$, $\delta=4$, $N=512$, 则 $|\Omega|=129032$. 如果采用全搜索, 则 $(512 \div 4)^2 = 16384$ 个 R 块共需约 211×10^7 次搜索! 这个简单的计算表明, 要使分形编码实用化, 还必须采用别的降低编码时间复杂度的方法.

二、块分类方案

(一) 视觉特性分类法

在分形图像编码的奠基性论文中, Jacquin^[1] 使用了一种根据几何视觉特性分类 D 块的方案, 该方案源于文献 [54] 的图像块分类方法. 在这种分类方法中, D 块被区分为三种不同的类型: 平滑块 (shade)、边缘块 (edge) 和中值块 (midrange), 它们的集合分别记为 \mathcal{D}_s 、 \mathcal{D}_e 、 \mathcal{D}_m , 即

$$\mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_e \cup \mathcal{D}_m \quad (8.8.1)$$

平滑块是亮度变化不明显的图像块, 边缘块是亮度急剧变化的图像块 (常常是含有边缘的图像块), 中值块的亮度变化介于平滑块和边缘块之间, 它们主要是包含纹理的图像块. 边缘块又进一步划分为两类——简单块 \mathcal{D}_{se} 与混合边缘块 \mathcal{D}_{me} (mixed

edge): $D_e = D_{se} \cup D_{me}$. 平滑块的亮度变化很小, 可以用固定的 DC 块的某个倍数来近似 (实际上是把对比度调整因子设置为 0), 因此避免了搜索其匹配块的过程. 于是, 可以从 D 中去掉平滑块 D_s , 即 $D = D_e \cup D_m$, 这样就减少了 D 中图像块的数目.

(二) 亮度均值和方差分类法

根据亮度均值和方差来分类是一种更精细的分类方法^[4,32,55]. 具体做法如下: 一个图像块 (R 块和 D 块) 被等分成四个子块 (左上、右上、左下和右下分别标号 1、2、3、4), 并分别计算每个子块的像素亮度平均值 A_i 和相应的亮度方差 V_i ($i=1,2,3,4$). 容易看出, 总可以通过重定向 (旋转或反转) 图像块 (R 块和 D 块) 使得亮度平均值 A_i 按下面的三种方式排序 (称为规范排序, canonical ordering):

$$\begin{aligned} \text{Class1: } A_1 &\geq A_2 \geq A_3 \geq A_4, \\ \text{Class2: } A_1 &\geq A_2 \geq A_4 \geq A_3, \\ \text{Class3: } A_1 &\geq A_4 \geq A_2 \geq A_3, \end{aligned} \quad (8.8.2)$$

于是, 根据亮度平均值的分布情况, 图像块 (R 块和 D 块) 被分为上面的 3 大类: R 块池 $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$; D 块池 $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$.

接着每一大类 \mathcal{R}_i (或 \mathcal{D}_i) 再按方差的 $4!=24$ 种排列分成 24 子类: $\mathcal{R}_i = \mathcal{R}_{i,1} \cup \mathcal{R}_{i,2} \cup \dots \cup \mathcal{R}_{i,24}$, $\mathcal{D}_i = \mathcal{D}_{i,1} \cup \mathcal{D}_{i,2} \cup \dots \cup \mathcal{D}_{i,24}$ ($i=1,2,3$). 这样就把所有的图像块 (R 块或 D 块) 分成了 $24 \times 3 = 72$ 子类:

$$\begin{aligned} \mathcal{R} &= \mathcal{R}_{1,1} \cup \mathcal{R}_{1,2} \cup \dots \cup \mathcal{R}_{1,24} \cup \mathcal{R}_{2,1} \dots \cup \mathcal{R}_{3,24}, \\ \mathcal{D} &= \mathcal{D}_{1,1} \cup \mathcal{D}_{1,2} \cup \dots \cup \mathcal{D}_{1,24} \cup \mathcal{D}_{2,1} \dots \cup \mathcal{D}_{3,24}, \end{aligned}$$

其中, \mathcal{R}_{ij} 与 \mathcal{D}_{ij} ($i=1,2,3; j=1,2,\dots,24$) 属于同一子类 (具有相同的均值与方差分布).

编码时, 给定一个 R 块 R , 其最佳匹配块 D 可以在四种不同的类中进行搜索:

(1) 1-类搜索 (one-class search) R 仅仅在与 R 和 $-R$ 具有相同规范类的方差子类中搜索匹配块 D . 详细说, 设 $R \in \mathcal{R}_{i,j}$, $-R \in \mathcal{R}_{k,l}$, 则仅仅在子类 $\mathcal{D}_{i,j} \cup \mathcal{D}_{k,l}$ 中搜索 R 的匹配块 D .

(2) 3-类搜索 (three-class search) R 仅仅在与 R 和 $-R$ 具有相同方差子类 (共 3 类, 分属 3 个规范类) 中搜索匹配块 D .

(3) 24-类搜索 (twenty-four-class search) R 仅仅在与 R 和 $-R$ 具有相同规范类 (每个规范类有 24 个方差子类) 中搜索匹配块 D .

(4) 72-类搜索 (seventy-two-class search) R 在所有的方差子类 (共 72 类) 中搜索匹配块 D , 即全搜索.

简言之, 编码时, 只有属于同一子类的 R 块和 D 块才进行比较. 此外, 只考虑把使 $t_k(R) \in \mathcal{R}_i$ 以及 $t_k(D) \in \mathcal{D}_i$ 的等度变换 $t_k, k \in \{0, 1, \dots, 7\}$.

这种分类方法尽管有效, 但缺点也是明显的, 因为对某个 R 块, 当在对应的 D 块子类中搜索不到匹配块时, 搜索不能扩展到相邻子类中. 特别地, 当对应的 D 块子类为空集时, 显然就会出现问题. 一种解决办法是图像块的方差排序被代之与灵活的方差向量排序^[56]; 另一种是采用聚类方法^[33].

(三) 结构分类法

分形编码中每个 R 块在码书 Ω 中的最佳匹配块中的“最佳”需要用某种度量来表征, 通常是 RMS(Root Mean Square) 或 MSE. 以典型的 8×8 固定方块分割为例, 计算每个 MSE 大约各需要 64 次减法和乘法运算. 如果能把某些 RMS 代之以计算复杂度低的度量, 自然可以降低编码计算复杂性. 为了实现这个想法, Hurtgen 和 Stiller 观察到这样的事实, R 块与其在码书 Ω 中的最佳匹配块具有近似的灰度级空间分布. 为此, 他们提出了基于局部均值的结构分类法^[57]. 具体做法如下: 与亮度和亮度方差分类方案类似, 一个图像块 R (R 和 D) 被等分成四个互不重叠的子块 $R_i (i=1, 2, 3, 4, \text{左上、右上、左下和右下分别标号 } 0, 1, 2, 3)$, 并计算图像块 R 的亮度平均值 m_R 和四个子块 R_i 的亮度平均值 m_i . 接着按下面的方法构造一个对应于 R 的特征向量 (feature) X_R :

$$X_R = (\chi_0, \chi_1, \chi_2, \chi_3), \text{ 其中 } \chi_i = \begin{cases} 1, & m_i > m_R, \\ 0, & m_i \leq m_R, \end{cases} \quad i = 0, 1, 2, 3. \quad (8.8.3)$$

对于两个图像块 R 和 D , 定义它们对应的特征向量的距离 (结构距离) 为

$$d_S(R, D) = \begin{cases} 0, & X_R = X_D, \\ \infty, & X_R \neq X_D. \end{cases} \quad (8.8.4)$$

这就是说, $d_S(R, D) = 0$ 当且仅当 $X_R = X_D$.

在搜索每个 R 块 R 在码书 Ω 中的最佳匹配块 D 的过程中, 只有 $d_S(R, D) = 0$ 的 R 和 D 才计算它们的 RMS, 从而避免了大量 RMS 的计算. 换言之, 对每个 R 块 R , 码书划分为两类 $\Omega = \Omega_R \cup (\Omega - \Omega_R)$, Ω_R 由满足 $d_S(R, D) = 0$ 的 $D \in \Omega$ 组成. 编码 R 块 R 时, 只需在码书子类 Ω_R 中搜索最佳匹配块. 当然, 必须事先计算 $d_S(R, D)$, 但计算这个结构距离的复杂性小于计算相应的 RMS.

我们也可以这样来看, 对于给定的码书 Ω , 可以按照每个码书块的特征向量把码书 Ω 分成 16 类: $\Omega = \Omega_0 \cup \Omega_1 \cup \Omega_2 \cup \dots \cup \Omega_{15}$ (例如 Ω_1 是特征向量为 $(0, 1, 1, 1)$ 的图像块组成的). 对每个 R 块 R , 计算其特征向量 X_R , 设为 $(0, 1, 1, 1)$, 则我们只需在码书 Ω_1 中搜索 R 块 R 的最佳匹配块. 因此, 本方案是一种块分类方案.

图 8-17 给出了图像块的特征向量的概率分布,它是通过对一系列测试图像的实验所得到的.图中底部的方块显示了图像块可能的灰度级空间分布, u 表示其类别.

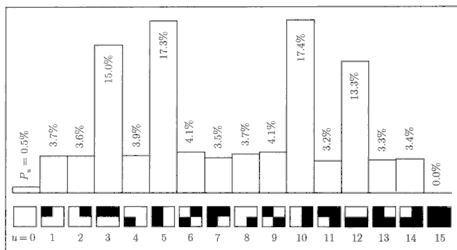


图 8-17 特征向量的概率分布

三、特征向量法

在分形图像处理技术中,图像块常常需要重新排序使其对应于一个线性空间 R^n 中的向量(矩阵向量化).也就是说,我们把 $N \times N$ 图像块 (N 阶方阵)看成线性空间 R^n 中的向量(列向量, $n = N \times N$),可以按多种向量化方式进行.假设坐标系原点位于图像的左上角,则图 8-18 给出了四种常用的图像块向量化方式.

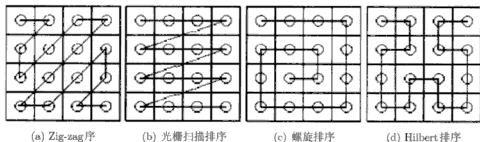


图 8-18 四种常用图像块向量化方式

用向量代替二维阵列(矩阵)既可以简化记号,又不失一般性,因此常常在分形图像编码及其他图像处理技术中采用.

下面我们约定码书 Ω 中的码书块都按某种方式向量化,即 $D \in \Omega$ 表示线性空

间 R^n 中的向量 ($n = N \times N, N \times N$ 是码书块 D 的尺寸).

Saupe^[58] 提出的特征向量法是一种有效的分形编码复杂度约简方法. 在这种方法中, 首先构造图像块 X (R 块和 D 块) 的特征向量 $\phi(X)$, 然后证明极小化 $MSE(R, D)$ 的问题等价于极小化

$$\Delta(R, D) \triangleq \min (d(\phi(R), \phi(D)), d(-\phi(R), \phi(D))) \quad (8.8.5)$$

的问题, 其中, d 是欧氏距离. 于是把“ R 在集合 $\{D\}$ 中搜索 MSE 意义下的最优匹配块”的问题转化为“特征向量 $\phi(R)$ 在特征向量空间 $\{\phi(D)\}$ 中搜索最优匹配块”的问题. 而后者实际上就是在 $2N$ 个 $\pm\phi(D)$ 的集合中搜索 $\phi(R)$ 的最近邻问题.

欧氏空间 R^n 中搜索最近邻的问题在计算机科学中得到了较为透彻的研究^[59], 例如, Friedman 等人 1977 年提出的 kd -tree 最近邻搜索就运行在对数时间下. 于是采用 kd -tree 最近邻搜索法 (或后来的改进算法), 特征向量空间中的最近邻法把搜索时间从全搜索时间 $O(N_R \cdot N_D)$ 减少为 $O((N_R + N_D) \log(N_D))$ (考虑了预处理时间), 其中 N_R, N_D 分别是 R 块和码书块的个数.

正如 Saupe 本人所指出的, 上述特征向量法有几个缺点 (文献 [60] 对此作了进一步的讨论). 例如, R 块和 D 块的特征向量的维数很高, 这样就存在特征向量的存储问题, 此外, 高维时 kd -tree 最近邻搜索方法的性能并不好 (维数越高, 性能越差). Saupe 提出的解决办法有两个: 一是通过像素平均或欠采样缩小 R 块和 D 块的维数 (如 4×4); 二是量化 R 块和 D 块特征向量的各分量 (8bit 足够). 这样处理的代价是搜索得到的特征向量 $\phi(R)$ 的最近邻只能是近似的. 此外, 即使是准确的, 但对应的尺度因子 s 未必满足需要的约束 (以保证分形变换的压缩性). 于是, Saupe 提出在搜索得到的特征向量 $\phi(R)$ 的最近邻附近再搜索 5~10 个近邻, 从中选择满足尺度因子约束且 $MSE(R, D)$ 最小者.

下面两节介绍作者提出的两个快速分形编码算法.

四、适应码本缩减方案

我们提出一个改进的分形编码方案——适应码本缩减算法. 该算法的基本原理是利用对比度因子与码块 (codebook block) 均方差的反比关系, 预先从码本中排除那些不太可能满足对比度因子约束 ($|s| < 1$) 的码块. 因此, 匹配搜索空间可以限制到仅仅由方差较大的码块组成的适应缩减码本, 既减小了对比度因子不满足约束的可能性, 又大大缩小了最佳匹配块的搜索范围. 仿真结果表明, 对于 10 幅复杂性不同的 8 比特量化 256×256 标准测试图像, 与对应的全搜索分形算法比较, 在 PSNR 度量的客观质量 (PSNR) 略有增加、主观质量无下降的前提下, 编码时间显著地减少 (平均加快约 10 倍). 作为比较, 在保持 PSNR 不变的条件下, 文献

[61] 的算法加快编码 2.7 倍, 文献 [62] 的加快倍数依赖于 R 块的尺寸: 对于 4×4 , 加快 0.56 倍; 对于 8×8 , 加快 2.15 倍. 此外, 本算法的理论分析非常简单, 算法也容易实现, 而文献 [61, 62] 却需要十分复杂的理论分析, 算法实现也复杂. 实际上, 按本方案, 只需把现有分形编码算法中的虚拟码本换成适应缩减码本, 不用修改其余部分. 因此, 目前所有分形编码程序只需作少许修改即可融入本方案. 顺便指出, 适应码本缩减算法是作者在论文 [63] 中提出的算法的进一步发展.

为了叙述简明且不失一般性, 我们讨论分形编码的基本类型, 并从矢量量化编码的观点叙述其算法. 我们知道, 在基本分形编码中, 图像被分割成大小两类子块: range 块 (R 块) 和 domain 块 (D 块), R 块互不重叠且覆盖整幅图像, D 块可以重叠且边长通常为 R 块边长的两倍. D 块经四邻域像素值平均收缩为 R 块的大小, 收缩后的 D 块按等距变换扩大为 8 个子块, 这种子块的全体就构成虚拟码书 (记为 Ω). 然后, 每个 R 块 R 由其最佳匹配块 $D \in \Omega$ 的亮度变换来近似, 即 $R \approx s \cdot D + o \cdot 1$, 其中 1 是亮度值均为 1 的常值块, s 、 o 分别起调整子块 D 的对比度和亮度的作用.

编码阶段, 对于每个 R 块 R_i , 为了寻求其最佳匹配块 $D_{m(i)} \in \Omega$, 我们必须求解下面的极小化问题 (把 R 、 D 视为向量):

$$\|R_i - (s_i \cdot D_{m(i)} + o_i \cdot 1)\| = \min_j \left\{ \min_{s, o \in \mathbb{R}, |s| < 1} \|R_i - (s \cdot D_j + o \cdot 1)\| \right\}, \quad (8.8.6)$$

其中 $\|\cdot\|$ 是向量 2-范数, $m(i)$ 、 s_i 和 o_i 分别表示 R_i 的最佳匹配块 $D_{m(i)} \in \Omega$ 的序号、对比度和亮度的最优调整因子. 应该注意, 对比度因子必须满足约束 $|s| < 1$ 以保证解码迭代序列收敛. 极小化问题 (8.8.6) 的三元组 $(m(i), \bar{s}_i, \bar{o}_i)$ 称为 R_i 的分形码, 其中 \bar{s}_i 、 \bar{o}_i 是量化值. 全体 R_i 的分形码就组成整幅图像的分形码, 它描述了一个使原始图像近似不变的压缩仿射变换 T . 解码是相对简单的迭代过程, 由 Banach 压缩映射原理给出的迭代算法完成, 即原始图像 μ_{org} 由分形码表示的压缩变换 T 迭代作用于任何初始图像 μ_0 来生成 $\mu_{\text{org}} \approx T^N(\mu_0)$. 拼贴定理保证变换不动点是原始图像的一个近似图像, 因此, 这种基于拼贴定理的分形编码也常常称为拼贴编码 (collage coding).

尽管拼贴定理目前是所有分形编码的基础, 但是拼贴编码并没有导出图像的最优分形表达 [3]. 这主要有下列原因:

(1) 为了寻求表达原始图像 μ_{org} 的压缩仿射变换 T , 应该极小化吸引子误差 $\|\mu_{\text{org}} - \mu_T\|$ 而不是拼贴误差 $\|\mu_{\text{org}} - T\mu_{\text{org}}\|$, 式中 μ_T 是变换 T 的不动点 (吸引子). 尽管拼贴定理指出吸引子误差不超过拼贴误差的 $(1-s)^{-1}$ 倍 (s 是变换 T 的压缩因子), 拼贴误差小可以保证吸引子误差也小, 但并不能保证吸引子误差小于拼贴误差 (因为 $(1-s)^{-1}$ 大于 1). 实际上, 许多实验表明 [3], 拼贴误差通常小于吸引子误差, 可见拼贴编码绝非最优编码. 此外, 已经证明 [48], 极小化吸引子误差的最优分

形编码是 NP- 难问题. 因此, 只能寻求分形编码的次优算法, 拼贴编码是最成功的次优算法, 成了分形编码事实上的标准算法.

(2) 为了用拼贴编码方法压缩一幅图像, 我们必须求解大量的交错极小化问题 (8.8.6); 也就是说, 对于原始图像的每个 R 块 R_i , 我们需要确定最优系数 s_i 、 o_i 和最佳匹配块序号 $m(i)$. 此外, 对比度因子必须满足约束 $|s| < 1$ 以保证解码迭代序列收敛. 不难看出, 这是十分复杂的约束优化问题, 计算非常费时. 为了计算简单可行, 在实用分形编码中, 问题 (8.8.6) 中内层约束极小化问题通常忽略对比度因子 s 的约束, 从而变成典型的最小平方问题, 于是可以采用最小二乘法简单求解. (8.8.6) 中外层极小化问题在基本分形编码中则采用全搜索方法求解. 忽略对比度因子约束的做法虽然有实验结果的支持: 对许多标准测试图像的实际计算表明, s 的值绝大多数分布在 $[-1, 1]$. 但是, s 的值并非全部分布在 $[-1, 1]$, 传统做法是对不满足约束的 s 作切断处理 (例如切断 s 为 0, 即把对应的 R 块人为当作常值块, 其亮度为其亮度均值). 这样做有两个缺点: 一是可能降低解码图像质量 (例如把一些细节丰富的块当作常值块), 二是浪费计算时间 (必须计算后才知道 s 是否满足约束).

因此, 针对上述第二个原因, 如果在编码过程中适当地把对比度因子约束考虑进去, 从码本中预先排除那些不太可能满足约束的码块, 图像质量和编码时间都应该会有所提高. 本节就是讨论如何减少上述第二个原因产生的质量下降和时间浪费问题.

算法理论分析. 给定一个 R 块 $R \in R^n$ 和一个码块 $D \in \Omega \subset R^n$, 设 $E(R, D)$ 是 R 与 D 的亮度变换版本 $a \cdot D + b \cdot \mathbf{1}$ 之间的最小均方误差:

$$E(R, D) = \frac{1}{n} \min_{a, b \in \mathbb{R}} \|R - (a \cdot D + b \cdot \mathbf{1})\|^2 = \frac{1}{n} \|R - s \cdot D - o \cdot \mathbf{1}\|^2. \quad (8.8.7)$$

显然, 如果忽略对参数 a 的约束 $|a| < 1$, 则问题 (8.8.6) 中内层约束极小化问题等价于最小平方问题 (8.8.7). 实际上, 问题 (8.8.7) 是变量 a, b 的二阶多项式的最小值问题, 通过直接而简单的计算, 我们得到

$$E(R, D) = \text{var}(R) - s^2 \cdot \text{var}(D), \quad (8.8.8)$$

$$s = \frac{\langle R - \bar{R} \cdot \mathbf{1}, D - \bar{D} \cdot \mathbf{1} \rangle}{n \cdot \text{var}(D)}, \quad o = \bar{R} - s \cdot \bar{D}, \quad (8.8.9)$$

其中, $\langle \cdot, \cdot \rangle$ 表示向量的欧氏内积, \bar{X} 和 $\text{var}(X)$ 分别是子块 X 的亮度均值和亮度均方差:

$$\bar{X} = \frac{1}{n} \langle X, \mathbf{1} \rangle, \text{var}(X) = \frac{1}{n} \|X - \bar{X} \cdot \mathbf{1}\|^2. \quad (8.8.10)$$

我们已在前面提到, 分形编码中的亮度仿射变换通常要求对比度因子满足约束 $|s| < 1$, 但这个约束在问题 (8.8.7) 中被人为地忽略. 尽管我们可以对不满足约

束的 s 作切断处理, 但这样做可能降低图像质量, 也可能浪费计算时间. 因此, 在分形编码中, 适当地考虑比度因子约束可能会得到更好的编码性能 (图像质量和编码时间).

从式 (8.8.9) 可知, 对比度因子 s 与码块 D 的均方差 $\text{var}(D)$ 成反比关系, 较小的均方差 $\text{var}(D)$ 可能伴随较大的 s 值 (不满足约束, 即 $|s| > 1$). D 的均方差越小, s 的值可能越大, 从而 s 不满足约束的可能性越大, 需要作切断处理的 s 越多, 从而导致解码图像质量的下降. 但是, s 表达式的分子也与 D 有关, 因此, D 的均方差 $\text{var}(X)$ 小, 如果 $\langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle$ 也很小, 理论上就不能保证 s 的值大 (不满足约束). 然而, 如果 $\langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle$ 很小, 则 R 、 D 的相关性小, 从而成为最好匹配对的可能性小, 于是出现上述情况的可能性小. 因此, 解码图像质量可能因预先排除小方差的码块而得到改进.

从式 (8.8.8) 可知, 对于任意码块 $D \in \Omega$, 成立 $E(R, D) \leq \text{var}(R)$. 于是

$$\text{var}(R) < \delta^2 \Rightarrow E(R, D) < \delta^2, \quad (8.8.11)$$

其中, $\delta > 0$ 是预设阈值. 换言之, 如果 $\text{var}(R)$ 小, 则 $E(R, D)$ 也小. 于是, 我们可以对 R 块进行简单的分类: 如果 $\text{var}(R)$ 小于预设阈值 (即 $\text{var}(R) < \varepsilon^2$), 则 R 称为平滑块 (亮度变化很小); 否则称为非平滑块. 显然, $\text{var}(R)$ 越小, 子块 R 越接近于常值块; 特别地, $\text{var}(R)$ 为零的子块 R 就是常值块. 因此, 平滑块 R 可以近似视为常值块, 其亮度值可以用平均亮度值代替: $R \approx \bar{R} \cdot 1$, 从而无需在码本中搜索其匹配块. 于是, 只有非平滑 R 块才需要在码本中搜索其匹配块.

再次从式 (8.8.8) 可知, $\text{var}(R)$ 与 $\text{var}(D)$ 不大可能相差太大, 因此, 对于具有较大方差的 R 块 (非平滑块), 其最佳匹配 D 块一般也可能具有较大的方差.

通过以上分析, 小方差的码块不太可能成为非平滑 R 块的匹配块, 同时也可能伴随较大的对比度因子 (不满足约束), 因此可以预先从码本中有效地排除. 于是, 我们定义适应缩减码本如下:

$$\Omega_\delta = \{D \in \Omega : \text{var}(D) \geq \delta^2\}, \quad (8.8.12)$$

其中, $\delta > 0$ 是预设阈值.

显然, 适应缩减码本 Ω_δ 比码本 Ω 小, 因此, 对于给定 R 块 R , 在适应缩减码本中搜索其最佳匹配块的范围缩小, 搜索量减少, 不难想见编码时间也会随之减少. 于是, 改进方案在解码图像质量和编码速度方面都应该有所提高. 尽管如此, 必须通过实验求证上述论断. 此外, 分别定义适应缩减码本的阈值 δ 和平滑块的阈值 ε , 理论上不能确定其最佳数值, 只能通过实验求取.

下面给出实验结果. 在改进算法中, 重构图像质量和编码时间依赖于两个新增控制参数: 平滑块阈值 ε 和适应缩减码本阈值 δ . 理论上不能确定其最佳数值, 只能通过实验求取. 计算在 AMD Duron/850MH CPU、运行 Windows 2000 的计

计算机上进行. 程序用 C++ 编写. 测试编解码性能的参数是源于操作系统时钟的编码时间 (秒) 和峰值信噪比 (PSNR). 测试对象是 10 幅具有不同复杂性的 256×256 标准测试图像 (8 bit/pixel), 它们是 “Lena”, “Goldhill”, “Bridge”, “Bird”, “Barb”, “Zelda”, “Baboon”, “San”, “Boat” 和 “Pepper” 图像 (图 8-19).

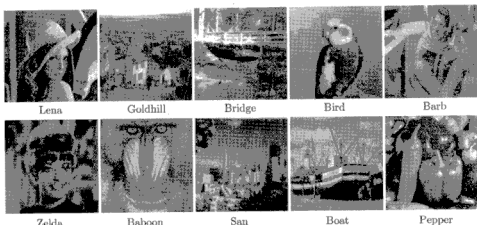


图 8-19 具有不同复杂性的测试图像

实验目标主要是验证这样的命题, 解码图像质量能够因预先排除可能伴随较大对比度因子 (不满足约束) 的小方差的码块而得到改进. 因此, 我们设置 $\varepsilon = 0$ 以比较改进算法与对应的全搜索算法的编码性能. 顺便指出, 我们对阈值 ε 的取值也进行了仿真研究. 我们在实验中选取 $\varepsilon = 0, 1, 2, \dots, 10$, 结果表明, 较好的编码性能在 $\varepsilon \in [5, 7]$ 达到.

对于适应缩减码本阈值 δ , 我们实验中选取 $\delta = 45, 40, \dots, 10, 5$, 实验结果在表 8-2 中给出. 表中最后一行记录了平均 PSNR 和平均编码时间, 它们分别是所选 10 幅图像在不同阈值下的 PSNR 和编码时间的算术平均值, 这一行的最后一格记录了用对应的全搜索算法编码同样的 10 幅图像的平均 PSNR 和平均编码时间. 此外, 表中最后一列记录了用对应的全搜索算法编码同样的 10 幅图像的 PSNR 和编码时间. 表 8-2 显示, 当 $\delta \leq 35$ 时, 对于前 8 幅图像, 改进算法的 PSNR 都大于对应的全搜索算法的 PSNR; 当 $\delta \leq 25$ 时, 对于后 2 幅图像, 改进算法的 PSNR 也大于对应的全搜索算法的 PSNR. 这个结果的确验证了命题——解码图像质量能够因预先排除可能伴随较大对比度因子 (不满足约束) 的小方差的码块而得到改进.

图 8-20 给出了不同阈值 ($\delta = 45, 40, \dots, 10, 5$) 下的平均 PSNR vs. 平均编码时间曲线. 该曲线显示, 当阈值 δ 从 45 减少到 25 时, 平均 PSNR 迅速增加; 当阈值 δ 从 20 减少到 5 时, 平均 PSNR 缓慢下降. 此外, 平均编码时间随着阈值 δ 的减少而迅速增加. 因此, 较好的编码性能 (PSNR 和时间) 在 $\delta \in [25, 35]$ 达到. 我

表 8-2 基本算法与改进算法的比较

δ		5	10	15	20	25	30	35	40	45	Baseline
Lena	PSNR	27.83	27.89	27.91	27.92	27.9	27.87	27.86	27.85	27.72	27.78
	Time	97.1	68.63	56.14	43.10	32.77	24.85	18.88	13.71	9.92	131.5
Goldhill	PSNR	26.84	27	27.14	27.26	27.26	27.18	27.09	26.9	26.77	26.83
	Time	109.39	76.67	50.38	29.49	17.21	11.46	8.2	5.43	3.52	131.6
Bridge	PSNR	25.42	25.45	25.51	25.59	25.59	25.56	25.42	25.19	24.98	25.4
	Time	129.4	110.3	80.4	53.54	35.69	24.8	16.63	10.15	7.28	131.7
Bird	PSNR	32.21	32.28	32.29	32.31	32.23	32.21	32.2	32.06	31.91	32.0
	Time	44.4	31.9	24.35	19.16	15.19	11.12	8.75	6	4.69	131.1
Barb	PSNR	28.22	28.28	28.38	28.39	28.47	28.44	28.31	28.05	27.68	28.19
	Time	104.39	77.52	57.74	42.83	31.2	21.83	14.21	8.7	4.94	132.1
Zelda	PSNR	29.57	29.67	29.76	29.8	29.78	29.74	29.62	29.29	29.29	29.52
	Time	109.37	71.39	42.81	21.65	12.45	6.91	4.62	2.79	2.79	131.7
Baboon	PSNR	24.36	24.38	24.46	24.68	24.82	24.75	24.46	24.07	23.89	24.36
	Time	125.18	103.68	72.11	38.31	19.27	10.1	5.38	3.44	2.28	131.8
San	PSNR	26.54	26.64	26.68	26.71	26.75	26.7	26.61	26.44	26.10	26.5
	Time	81.49	69.39	57.86	43.97	34.34	26.35	16.87	12.32	8.42	131.4
Boat	PSNR	28.07	28.11	28.11	28.11	28.05	28.01	27.89	27.66	27.67	28.04
	Time	88.2	74.91	52.57	43.39	34.67	25.95	18.93	13.15	7.88	131.4
Pepper	PSNR	29.14	29.2	29.2	29.18	29.12	29.02	29	28.77	28.62	29.07
	Time	96.81	71.57	55.92	45.5	36.29	28.53	24.26	17.16	13.93	131.3
Mean value	PSNR	27.82	27.89	27.94	28	28	27.95	27.85	27.63	27.46	27.77
	Time	98.57	75.6	55.03	38.09	26.91	22.04	13.67	9.28	6.56	131.56

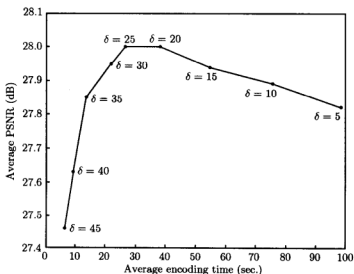


图 8-20 平均 PSNR vs. 平均编码时间曲线。其中，平均 PSNR 和平均编码时间分别是所选 10 幅图像在不同阈值下的 PSNR 和编码时间的算术平均值。此外，用对应的全搜索算法编码同样的 10 幅图像的平均 PSNR 和平均编码时间分别约为 27.77dB 和 131.6 秒

们推荐阈值 δ 的取值为 35, 因为在这种情况下, 与对应的全搜索算法相比, 改进算法在编码速度提升约 10 倍的情况下, PSNR 平均增加 0.08dB.

然而, PSNR 有时并不是图像质量的完美度量, PSNR 的增加未必伴随图像主观质量的改善, 反之亦然. 因此, 给出用改进算法与对应的全搜索算法分别编码所选 10 幅图像的主观质量对比是必要的. 但是, 篇幅所限, 这里仅仅给出 3 幅图像的解码图像对比 (图 8-21~ 图 8-23), 不难看出, 与对应的全搜索算法的解码图像对比, 改进算法的解码图像的主观质量几乎没有下降, 而编码时间却大大地减少.



图 8-21 Zelda 解码图像: 基本分形编码 (左); 改进算法 ($\delta = 35$)(右), PSNR: +0.12 dB, 加快: 24 倍

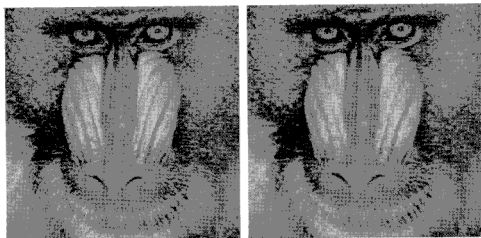


图 8-22 Baboon 解码图像: 基本分形编码 (左); 改进算法 ($\delta = 35$)(右), PSNR: +0.10 dB, 加快: 28 倍



图 8-23 Goldhill 解码图像: 基本分形编码 (左); 改进算法 ($\delta = 35$) (右), PSNR: +0.26 dB, 加快: 16 倍

五、基于方差的快速算法

本节介绍作者提出的基于局部方差的快速分形算法^[64], 该算法推广并改进了文献 [65] 的算法, 编码结果也好于文献 [66] 的算法 (也是文献 [65] 算法的改进). 实验结果显示, 按照我们的改进算法, 编码时间能够极大地减少, 同时解码图像的 PSNR 甚至高于对应的全搜索分形算法. 此外, 在 PSNR 同为 27dB 时, 我们的算法比文献 [65] 提出的算法快 28 倍 (图 8-24).

Lee 等^[65] 提出了一个有效的基于图像块方差的快速编码算法. 该算法基于这样的命题, 两个等尺寸的图像块只有方差接近才可能组成匹配对 (尽管他们没有说明命题为真的原因, 但我们给出了命题成立的理论依据). 为了减少匹配搜索的空间, 依据上述命题, 他们按照码块方差大小赋予码本序结构, 于是所有潜在的最好匹配搜索被限制在相对小的搜索窗内. 通过选择适当的搜索窗大小, 该算法在解码图像降质很小的条件下加快编码 10 倍左右^[65]. Lai 等^[66] 改进了这个快速算法: 与原算法不同, 为了搜索 R 块的最好匹配块, 原算法的对称搜索窗被代之以非对称搜索窗. 也就是说, 原算法在与 R 块方差最接近的码块为中心的对称搜索窗中搜索 R 块的最好匹配码块, 而改进算法仅仅在与 R 块方差最接近的码块为中心的对称搜索窗的右半部分 (设码本按方差升序排列) 中搜索 R 块的最好匹配码块. 实验结果表明, 在解码图像质量略有下降的条件下, 改进算法比原算法加快约 50% 左右^[66]. 我们也追随文献 [65] 提出的方差算法, 新算法引进了两个参数以控制编码速度和解码图像质量. 下面详细介绍我们的改进算法^[64].

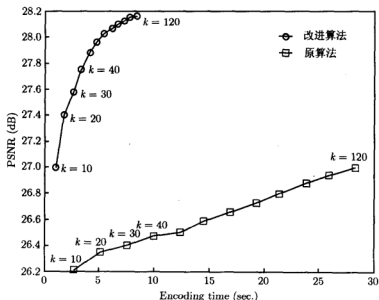


图 8-24 文献 [65] 与本文算法 ($\epsilon = 6$ 和 $\eta = 30$) 性能比较. 在给定搜索窗大小的 PSNR 与编码时间分别是所选 10 幅图像的 PSNR 和编码时间的算术平均值

给定一个 R 块 $R \in \mathbb{R}^n$ 和一个码块 $D \in \Omega \subset \mathbb{R}^n$, 设 $E(R, D)$ 是 R 与 D 的亮度变换版本 $a \cdot D + b \cdot 1$ 之间的最小均方根误差:

$$E(R, D) = \frac{1}{\sqrt{n}} \min_{a, b \in \mathbb{R}} \|R - (a \cdot D + b \cdot 1)\| = \frac{1}{\sqrt{n}} \|R - s \cdot D - o \cdot 1\|. \quad (8.8.13)$$

下面的等式是改进算法的理论基础, 能够通过简单的计算得到:

$$E(R, D)^2 = \text{var}(R) - s^2 \cdot \text{var}(D), \quad s = \frac{\langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle}{n \cdot \text{var}(D)}, \quad o = \bar{R} - s \cdot \bar{D}, \quad (8.8.14)$$

其中, $\langle \cdot, \cdot \rangle$ 表示向量内积, \bar{X} 是子块 X 的亮度均值, $\text{var}(X)$ 是定义为 $\text{var}(X) = \|X - \bar{X} \cdot 1\|^2 / n$ 的亮度方差.

从式 (8.8.14) 可知, 对于任意码块 $D \in \Omega$, 成立 $E(R, D)^2 \leq \text{var}(R)$. 这表明, 如果亮度方差 $\text{var}(R)$ 小, 则最小均方根误差 $E(R, D)$ 也小. 因此, 本文算法利用一个简单的分类方案^[1]于 R 块池以加快编码过程. 所有 R 块按照其亮度的变化情况分成两类: 如果亮度方差小于预定阈值, 则一个 R 块被看成阴影块 (shade block), 否则视为非阴影块 (non-shade block). 分类以后, 阴影块中的所有亮度值由亮度均值代替, 因为阴影块已经由常值块较好地逼近, 故不必再搜索码本. 此外, 式 (8.8.14)

也蕴涵一个 R 块的方差与其最佳匹配块的方差不大可能相差太大, 因此, 一个 R 块的最佳匹配块应该是这个 R 块在方差意义下的近邻. 此外, 在分形编码中, 亮度仿射变换实际上需要对比度因子满足约束 $|s| < 1$, 但这个约束在式 (8.8.13) 中没有被考虑. 传统做法通常对不满足约束的 s 作切断处理, 但这样做可能降低图像质量, 也可能浪费计算时间. 因此, 在编码中, 如果适当地考虑比度因子约束, 我们可能会得到更好的解码图像质量和更快的编码速度. 于是, 仅仅考虑码本中方差较大的码块应该是合理的, 因为具有较大方差的码块可能对应于小的对比度 s (见式 (8.8.14)).

以上分析表明, 更好的解码图像质量和更快的编码速度也许能够进一步获得, 如果把匹配搜索限制在下面定义的缩减码本内:

$$\Omega_\delta = \{D \in \Omega : \text{var}(D) \geq \delta^2\}, \quad (8.8.15)$$

其中, $\delta > 0$ 是预设阈值. 预先从码本中排除那些不太可能有用的码块以加快编码过程的思想类似于块分类, 但稍微有所不同. 在块分类中, 码本的每个子类都是有用的, 即每个 R 块都将在同类的码本子类中搜索其最佳匹配块.

因为每个非平滑块 R 块 R 与其最佳匹配块 $D \in \Omega_\delta$ 在方差意义下是近邻, 因此, R 的最佳匹配块 D 必须落入 Ω_δ 的下述子集:

$$\{D \in \Omega_\delta | |\text{var}(D) - \text{var}(R)| < \eta\}, \quad (8.8.16)$$

其中, $\eta > 0$ 是预设阈值. 如果容许码本 Ω_δ 按方差大小排序, 并设 D_m 是方差最接近非平滑 R 块方差的码块, 那么, 非平滑块 R 块的最佳匹配块 $D \in \Omega_\delta$ 也是 D_m 在方差意义下的近邻. 于是, 编码器需要对 D_m 在方差意义下的 k 邻域进行搜索以进一步改进质量. 此外, 在改进算法中, 码本 Ω 仅仅由收缩 D 块组成 (即没有作用 8 个等距变换), 于是, 为了进一步改进图像质量, 在 D_m 的 k 邻域中找到最佳匹配块 $D \in \Omega_\delta$ 后, 编码器还需要比较 D 的 8 个等距变换版本与 R 块的均方根, 具有最小均方根的 D 就是 R 块的最佳匹配块.

从理论上讲, 本文算法不能保证对于每个 R 块总能找到最佳匹配块, 但实验结果表明, 它是分形编码在匹配精度与计算复杂性之间一个较好的折中算法. 应该指出, 文献 [65] 的算法仅仅是改进算法的一个特例 (即参数 ε 和 η 都取 0). 此外, 为了独立显示本文算法的效果, 我们没有考虑文献 [66] 的非对称窗策略.

实验结果. 改进算法的实现采用分形编码的基本类型, R 块的大小取为 4×4 , 纵横方向步长取为 8 个像素点. 为了评价本文算法, 我们用全搜索分形算法、文献 [65] 的算法与改进算法分别编码 10 幅复杂性不同的图像. 这 10 幅图像是 256×256 “Lena”, “Barb”, “Zelda”, “Camera”, “Bird”, “Goldhill”, “San”, “Bridge”, “Boat” 和 “Pepper” (图 8-19). 重构图像质量由 PSNR 度量, 编码时间源于操作系统时钟. 所

有实验都在配置 AMD/Duron 850 MHz CPU 并运行 Windows 2000 的计算机上进行。

本文算法有三个参数影响重构图像质量与编码时间,即阴影块阈值 ε 、缩减码本阈值 δ 和搜索窗大小 k 。我们通过改变参数值 ($\varepsilon = 0, 1, 2, \dots, 10, \delta = 0, 5, 10, \dots, 50$) 以实验研究前两个参数的取值。结果显示,较高的参数值对应较快的编码但较低的图像质量,同时较低的参数值对应较慢的编码但较高的图像质量。因此,我们设置 $\varepsilon = 6, \eta = 30$ 以比较文献 [65] 算法与本文算法的性能。

图 8-24 给出了在不同搜索窗大小 ($k = 10, 20, \dots, 120$) 时 PSNR- 编码时间曲线,其中文献 [65] 的结果用小方块表示,改进算法 ($\varepsilon = 6, \eta = 30$) 的结果用小圆圈表示。PSNR 与编码时间是指平均 PSNR 和平均编码时间,它们分别是所选 10 幅图像在不同搜索窗大小时的 PSNR 和编码时间的算术平均值。此外,对应的全搜索算法编码同样的 10 幅图像的 PSNR 和编码时间分别是 28dB 和 131.5 秒。它显示平均编码时间显著地减少,同时与全搜索算法比较,平均 PSNR 反而有所增加。此外,在 PSNR 同为 27dB 时,本文算法比文献 [65] 提出的算法快 28 倍。

因为极快的编码速度但较低的图像质量可以在参数 ε 、 δ 取较高的值和搜索窗大小 k 取较低的值时获得,因此,本文算法在某些应用中有较好的应用前景,例如,对要求快速编码且图像质量要求不高的应用场合。

第九节 混合分形编码

分形图像编码有压缩比很高、分辨率无关性和解码速度快等公认的优点,但大多数纯粹基于分形理论的编码方法与目前广泛使用的编码方法相比,还缺乏竞争力。尽管如此,分形编码与其他编码方法(如小波、变换编码、矢量量化、预测编码等)结合的混合编码方法已经取得巨大成功。

目前提出的混合分形编码算法非常多,这里仅仅简单介绍作者认为有代表性的几个方法的编码思想,不涉及编码细节。

一、小波与分形编码

分形编码的多分辨分析是分形编码理论中最引人瞩目的成就,它不仅能够导致分形编码器性能的改进,而且也使我们更好地理解分形编码的内在机理^[67]。小波与分形图像编码的联系由许多学者独立发现^[3]。

我们知道,多分辨分析^[68]把小波的构造纳入统一的框架,同时也较好地指示了小波变换的显微能力。多分辨分析把信号(含图像)分解为多种尺度分量,并相应采用粗细不同的时域(空域)取样步长,从而能够不断地聚集于任意微小的细节。这种思想与我们对分形的观察与认识是一致的,体现了我们识别景物的渐进过程。例

如,由远及近地观察一处海岸景色,我们会首先注意到作为景色最显著特征的轮廓(海岸线),再慢慢注意其结构(如礁石),最后逐步观察其细节或纹理特征(如绿草).这种识别过程充分体现了一种从低分辨率到高分辨的原理.对于一般分形,通过从大到小不同尺度的变换,我们可以在越来越小的尺度上观察越来越丰富的细节.尺度不变性(scale-invariant)既是小波的特征,又是分形的特征.因此,作为分析工具的小波分析与作为几何语言的分形理论具有深刻的内在联系,它们在尺度变换上具有惊人的一致性.由此可见,对源于分形理论的分形图像编码进行小波分析就是自然而然的想法.

在分形编码中,图像分割后,尺寸为 $2^r \times 2^r$ 的子块通常由另一块尺寸为 $2^{r+1} \times 2^{r+1}$ 的子块的某个仿射变换版本来近似,这实际上是在图像中寻找不同尺度下的相似结构.对分形编码进行多分辨率分析的第一个方法是 Barahav 等人^[69]给出的分形编码的分级解释.下面,我们给出一个阐明这种分级思想例子.给定一幅 256×256 灰度图像,用固定方块分割方法把它划分为一系列不重叠的 8×8 子块,这样的子块共有1024个.按照前一章的基本分形编码算法对它进行编码,分形码由1024个仿射变换的系数、D块位置信息和等距变换序号组成.解码时,分形码表示的压缩变换可以迭代生成 256×256 吸引子图像 A_1 (按 8×8 分割初始图像),也可以迭代生成 128×128 吸引子图像 A_2 (按 4×4 分割初始图像)、 64×64 吸引子图像 A_3 (按 2×2 分割初始图像)和 32×32 吸引子图像 A_4 (按 1×1 分割初始图像).

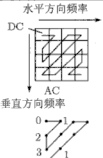






用小波分析方法显式地描述分形编码由许多学者独立完成,如 Davis、Walle、Simon 等人,读者可以从综述文献[3]中所列参考文献找到相关资料(Davis 的另一篇论文请参看文献[67]).此外,文献[70]提出了混合分形零树(zerotree)小波编码方法.

二、变换与分形编码

Barthel 等在文献[71]中提出了频域上的适应分形编码.他们把通常的亮度变换 $a \cdot D + b \cdot 1$ 修改为 $a \cdot (D - \bar{D} \cdot 1) + 0.5 \bar{D} \cdot 1 + b \cdot 1$ (其中 \bar{D} 表示D的亮度均值),并把修改后的亮度变换从空域转化到频域上去(通过DCT变换),同时采用了一种适应搜索方法寻找R块的最佳匹配块.实验结果表明,对于 512×512 Lena 图像,这种编码方法得到了很好的结果:在压缩比80:1下,PSNR=30dB;此外,在压缩比13:1~80:1下,PSNR都高于JPEG编码.随后,Barthel在文献[72,73]中提出了统一分形编码与变换编码的分形变换编码(fractal transform coding),它能够探索图像中的块间(inter-block)冗余(通过分形编码),又能够探索图像中的块内(intra-block)冗余(通过变换编码).编码在频域上进行,R块的大多数频谱系数用分形编码,只有那些不能用分形变换得到较好逼近的频谱系数才使用变换编码.因此,这种混合编码方法实现了更好的编码效率,在各种压缩比下,编码结果

都优于 JPEG 编码. 表 8-3 给出了比较频域上的分形块编码、分形变换编码与变换编码的例子, 其中, \hat{F} 是 R 块的离散余弦变换 (DCT), 并按 zig-zag 序向量化. 在表 8-3 中第三列给出的分形变换编码中, R 块的频谱系数有两个用变换编码, 其余用分形编码.

表 8-3 分形块编码、分形变换编码与变换编码的比较

编码方案	分形编码	分形变换编码	变换编码
Range 块的近似	$\hat{F} = \begin{bmatrix} a \cdot G_0 \\ a \cdot G_1 \\ a \cdot G_2 \\ \vdots \\ a \cdot G_i \\ \vdots \\ a \cdot G_{N^2-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}$	$\hat{F} = \begin{bmatrix} 0.5 \cdot G_0 \\ 0 \\ a \cdot G_2 \\ 0 \\ a \cdot G_i \\ \vdots \\ a \cdot G_{N^2-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ 0 \\ b_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$	$\hat{F} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_{N^2-1} \end{bmatrix}$
 <p>水平方向频率</p> <p>垂直方向频率</p>	<p>对比度因子</p>   <p>亮度偏移</p>	<p>对比度因子</p>   <p>亮度偏移</p>	<p>对比度因子</p>   <p>亮度偏移</p>

下面我们介绍上述三文采用的适应搜索方法, 因为它也可以用于其他的分形编码方案. 对于给定的 R 块, 他们通过实验研究了其最佳匹配块在码本中的位置分布, 结果显示, 最佳匹配块常常直接覆盖对应的 R 块或在对应的 R 块的附近. 据此实验结果, 他们提出了一个优化码本搜索路径. 对于给定的 R 块, 搜索始于直接覆盖它的 D 块, 然后采用螺旋路径搜索码本 (如图 8-25 所示). 当然, 这需要规定一种终止判据. 为此, Barthel 等引入“搜索区域”(search regions) 以实现可变搜索路径长度. 例如, 图 8-25 中的最小搜索区域由 16 个 D 块组成 (图中用小黑圆点表示). 另外, 还可以预设一个误差阈值. 于是, 对于给定的 R 块, 搜索始于直接覆盖它的 D 块, 并采用螺旋路径搜索码本, 如果逼近误差小于预设阈值或者搜索已经到达搜索路径的末端, 则搜索中止.

Curtis 等在文献 [74] 提出了另一个结合 DCT 的混合分形编码算法. 该文采用文献 [75] 提出的剪枝离散余弦变换 (Pruned DCT), 例如, 3 级 Pruned DCT 就是仅仅保留左上角的 6 个频谱系数 (按 zig-zag 序), 其余频谱系数都被剪枝为 0. 这种

混合方法的编码过程大致如下: 图像分割后, 对于某个 R 块 R , 首先进行离散余弦变换 $DCT(R)$, 并对它进行 3 级剪枝, 然后对剪枝后的版本进行 DCT 逆变换, 设结果为 \hat{R} . 因为 R 的中高频谱系数被剪枝, 因此 $\hat{R} \approx R$, 其误差由下式确定:

$$m(R, \hat{R}) = \sum_{i,j=0}^{N-1} \left(R(i, j) - \hat{R}(i, j) \right)^2, \quad (8.9.1)$$

其中 N 是 R 的行(列)数. 指定一个误差阈值 ε (Curtis 等建议取 500), 如果 $m(R, \hat{R}) < \varepsilon$, 则 R 用 DCT 编码, 否则试用分形编码, 设分形解码后的结果为 R' . 如果 $m(R, R') < \varepsilon$, 则 R 采用分形编码, 否则比较两种编码的误差 $m(R, \hat{R})$ 和 $m(R, R')$: 如果 $m(R, \hat{R}) > m(R, R')$, 则 R 采用分形编码, 否则采用 DCT 编码. 实验结果表明, 这种混合编码的性能大大优于对应的分形编码和剪枝 DCT 编码.

结合 DCT 的混合分形编码还有一些其他算法, 可以在文献 [76~83] 中找到详细资料. 此外, 文献 [84] 提出了结合 Hadamard 变换的快速分形编码方案.

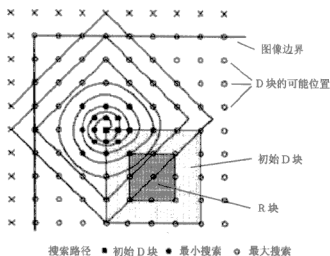


图 8-25 使用最小搜索区域的适应搜索方案

三、矢量量化与分形编码

在分形编码的奠基性论文 [1] 以及随后的综述文章 [40] 中, Jacquin 指出了分形编码与矢量量化编码的密切联系, 但没有对此作进一步的讨论. 之后, 许多学者对此进行了积极的探讨, 这里我们介绍三个有代表性的方法.

在文献 [85] 提出的算法中, 输入图像的低频分量采用 VQ 编码, 误差图像则采用分形编码. 具体地说, 先用四叉树分割方法 (基于分形维数) 把输入图像 $\mu(i, j)$ 划分为一系列不同尺寸的 R 块: $\mu = \cup_i R_i$. 对每个 R 块 R_i 进行 DCT 变换, 然后

用 VQ 方法量化其 DCT 版本 (仅仅留下低频系数), 接着对 VQ 量化后的 DCT 版本进行逆 DCT(IDCT) 返回空域, 设结果为 R'_i , 按四叉树分割信息“拼贴”出原始图像的近似图像 $\mu' = \cup_i R'_i$, 最后对误差图像 $\Delta\mu = \mu - \mu'$ 进行分形编码。

文献 [86] 研究了 VQ 和分形编码的联系, 即分形编码中的 D-R 映射 (domain-range mapping) 可以解释为双阶段 VQ (two-stage VQ), 并据此提出了一个分形矢量量化 (FVQ, fractal vector quantization) 编码方案, 推广了文献 [85] 的编码方案。在 FVQ 编码中, 原始图像由一组固定基块 (fixed basis block) 的线性组合粗略近似 (称为平面近似, planar approximation), 码本生成于这个粗略近似图像, 而不是其他训练图像或原始图像。下面, 我们稍微详细地讨论 FVQ 的算法步骤。

先用四叉树数据结构把原始图像分割成最大尺寸为 32×32 、最小尺寸为 4×4 的一系列 R 块, 每个尺寸不同的 R 块采用不同数量的固定基块来近似。在该文献中, 尺寸为 32×32 、 16×16 的 R 块由正交基块 X 、 Y 和 B 近似, 而尺寸为 8×8 和 4×4 的 R 块则仅仅由一个固定基块 B 近似。其中, 基块 X 、 Y 在 x 、 y 方向都是线性增加的, B 是亮度值均为 1 的常值块。于是, 原始图像的最佳平面近似 $\mu^p = \cup_i R_i^p$ 由下式确定:

$$R_i^p = \begin{cases} \gamma_i \cdot B, & 4 \times 4, 8 \times 8 \text{ } R_i, \\ \alpha_i \cdot X + \beta_i \cdot Y + \gamma_i \cdot B, & 16 \times 16, 32 \times 32 \text{ } R_i, \end{cases} \quad (8.9.2)$$

其中, $\langle A, B \rangle$ 表示 A 、 B 的内积, 且

$$\alpha_i = \frac{\langle X, R_i \rangle}{\langle X, X \rangle}, \quad \beta_i = \frac{\langle Y, R_i \rangle}{\langle Y, Y \rangle}, \quad \gamma_i = \frac{\langle B, R_i \rangle}{\langle B, B \rangle}. \quad (8.9.3)$$

接着是码本生成与优化。在平面近似中, 光滑区域由大平面 (large plane) 近似, 并被分割为大 R 块, 同时含有边缘的活性区域 (active region) 被分割为小 R 块。为方便起见, 该文用 4×4 和 8×8 块表示活性 R 块, 16×16 和 32×32 块表示光滑 R 块。为了生成码本, FVQ 编码器分别使用空域压缩和平滑算子把原始图像的平面近似图像变换成图像 Γ_1 和 Γ_2 。具体说, Γ_1 是原始图像的平面近似图像经 4×4 像素值平均得到的, 如 512×512 平面近似图像对应的 Γ_1 是 128×128 图像; Γ_2 是原始图像的平面近似图像经 9×9 移动平均滤波器 (moving average filter) 所生成的同尺寸平滑图像。设 C_r 表示 $r \times r$ 码本块的集合, 则 C_4 和 C_8 生成于 Γ_1 图像, 它们具有丰富的细节, 适合于编码活性 R 块; C_{16} 和 C_{32} 生成于 Γ_2 图像, 它们通过了低通滤波, 适合于编码光滑 R 块。然而, C_4 和 C_8 中的码向量有一些可能是相似的, 这种冗余在 FVQ 编码中能够被消除而无需附带另外的信息 (side information)。详细地, 设 C_4 或 C_8 为 $C = \{C_1, C_2, \dots, C_s\}$, 其中, 每个 C_i 都被规范化为零均值、单位方差的子块 (为记号简单, 仍用 C_i 表示)。C 中的冗余通过一个简单缩减算法消除: 对于 $i, j (1 \leq i < j \leq s)$, 如果 $\|C_i - C_j\|$ 或 $\|C_i + C_j\|$ 小

于预定阈值, 则从 C 中移去 C_j . 此外, C_{16} 和 C_{32} 生成于 Γ_2 图像, Γ_2 图像与原始图像是高度相关的, 因此, 对于每个 16×16 和 32×32 R 块, C_{16} 和 C_{32} 仅仅由位于 R 块附近的 25 个子块组成. 通过这种方法, C 被大大地缩减, 从而减少了编码时间.

最后是 R 块的精细量化 (fine quantization). 设尺寸为 $r \times r$ 的 R 块由 $C = \{C_n; 1 \leq n \leq s\}$ 量化为 $R_{i,n}$, 则

$$R_{i,n} = \begin{cases} \delta_{i,n} \cdot C_n + \gamma_i \cdot B, & 4 \times 4, 8 \times 8, \\ \delta_{i,n} \cdot C_n + \alpha_i \cdot X + \beta_i \cdot Y + \gamma_i \cdot B, & 16 \times 16, 32 \times 32, \end{cases} \quad (8.9.4)$$

其中, X 、 Y 和 B 属于 C , $\alpha_i, \beta_i, \gamma_i$ 由 (8.9.3) 计算, 而增益常数 $\delta_{i,n}$ 则通过 R_i 向由 $C = \{C_n; 1 \leq n \leq s\}$ 张成的子空间 $\text{span}\{C_n; 1 \leq n \leq s\}$ 作投影来计算.

文献 [85,86] 提出的编码算法的一个共同特点是, 与传统分形编码不同, 它们都是非迭代解码, 因此无需要求图像变换是压缩的. 顺便指出, 非迭代解码的分形编码方案始见于文献 [49].

文献 [87] 以一种自然而有效的方式把分形编码与 MSGVQ(mean shape-gain VQ) 联系起来, 并结合了该文作者以前的数篇论文的编码方案, 如基于聚类 (clustering-based) 的快速编码方案等. 该文较长, 内容较多, 对此有兴趣的读者请参看原文.

四、其他混合分形编码

目前提出的混合分形编码算法很多, 我们不再一一介绍了. 这里仅仅列举几个有代表性的论文, 如文献 [88,89](结合预测编码), 文献 [90](基于部分分形映射的混合编码方法), 文献 [91](结合神经网络), 文献 [92](结合遗传算法), 等等.

第十节 本章小结

在分形编码基本原理的框架内, 在提高编码质量和加快分形编码等方面, 许多新观点、改进算法以及混合分形编码算法被提出, 本章详细介绍了这些内容的主要研究概况. 目前已提出的分形编码方案非常多, 囿于篇幅和作者的水平, 我们主要介绍了其中发表于 IEEE/IEE 刊物、IEEE/IEE 国际会议论文集以及 Elsevier Science 刊物上的绝大部分论文的内容. 对于发表于国外其他刊物和国内刊物上的论文 (含作者的其他论文, 如 [93~98]), 本章基本没有介绍. 主要原因是, 对于发表于国外其他刊物的论文, 由于条件的限制, 我们无法收集到论文全文; 对于发表于国内刊物上的论文 (不乏优秀的论文), 考虑到读者很容易找到.

参 考 文 献

- [1] Jacquin A E. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1992, 1(1): 18~30
- [2] Wallace G K. The JPEG still picture compression standard. *Commun. ACM*, 1991, 34(4): 31~34
- [3] Wohlberg B, de Jager G. A Review of the Fractal Image Coding Literature. *IEEE Transactions on Image Processing*, 1999, 8(12): 1716~1729
- [4] Jacobs E W, Fisher Y, Boss R D. Image compression: A study of iterated transform method, *Signal Processing*, 1992, 29: 251~263
- [5] Saupe D, Jacob S. Variance-based quadrees in fractal image compression. *IEE Electronics Letters*, 1997, 33(1): 46~48
- [6] Shusterman E, Feder M. Image compression via improved quadtree decomposition algorithms. *IEEE Transactions on Image Processing*, 1994, 3(2)
- [7] Sullivan G J, Baker R L. Efficient quadtree coding of images and video. *IEEE Transaction on Image Processing*, 1994, 3(5): 327~333
- [8] Lu N. *Fractal Imaging*. New York: Academic, 1997
- [9] Hamzaoui R, Saupe D. Combining fractal image compression and vector quantization. *IEEE Transaction on Image Processing*, 2000, 9(2): 197~208
- [10] Melnikov G, Katsaggelos A K. A non uniform segmentation optimal hybrid fractal/DCT image compression algorithm. in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, Seattle, WA, 1998
- [11] Fuchigami T, Yano S, Komatsu T, Saito T. Fractal block coding with cost-based partitioning. in *Proc. Int. Picture Coding Symp.*, Melbourne, Australia, 1996, 335~340
- [12] Fisher Y, Menlove S. Fractal encoding with HV partitions. in *Fractal Image Compression: Theory and Application*. Fisher Y(Ed.). New York: Springer, 1994
- [13] Saupe D, Ruhl M, Hamzaoui R, Grandi L, Marini D. Optimal hierarchical partitions for fractal image compression, in *Proceedings ICIP-98 (IEEE International Conference Image Processing)*, Chicago, IL, USA, Oct. 1998, 1: 737~741
- [14] Wu X, Yao C. Image coding by adaptive tree-structured segmentation. in *Proc. IEEE Data Compression Conf.*, Storer J A, Reif J H(Eds.), Snowbird, UT, 1991, 73~82
- [15] Reusens E. Partitioning complexity issue for iterated function systems based image coding. in *Proc. VIIth Eur. Signal Processing Conf. EUSIPCO*, Edinburgh, U.K, 1994, 1: 171~174
- [16] Peitgen O, Jurgens H, Saupe D. *Chaos and Fractals New Frontiers of Science*. New York: Springer, 1992
- [17] Novak M. Attractor coding of images. In *Proceedings of the International Picture Coding Symposium PCS'93*, Lausanne, March 1993, 15~16
- [18] Davoine F, Svensson J, Chassery J M. A mixed triangular and quadrilateral partition for fractal image coding. in *Proc. IEEE Int. Conf. on Image Processing*, 1995, Washington DC, 3: 284~287
- [19] Jackson D J, Mahmoud W, Stepleton W A, Gaughan R T. Faster fractal image compression using recomposition. *Image Vision Computing*, 1997, 15: 759~767
- [20] Kumar S, Jain R C. Low complexity fractal-based image compression technique, *IEEE Trans on Consumer Electronics*, 1997, 43(4): 987~993

- [21] Davoine F, Chassery J M. Adaptive Delaunay triangulation for attractor image coding. in Proc. 12th Int. Conf. Pattern Recognition, Jerusalem, Israel, 1994
- [22] Davoine F, Robert G, Chassery J M. How to improve pixel-based fractal image coding with adaptive partitions. in *Fractals in Engineering*, V  hel J L, Lutton E, Tricot C (Eds.), U.K., London: Springer-Verlag, 1997, 292~306
- [23] Thomas L, Deravi F. Region-based fractal image compression using heuristic search. *IEEE Transactions on Image Processing*, 1995, 4(6): 832~838
- [24] Saupe D, Ruhl M. Evolutionary fractal image compression. in Proc. ICIIP-96 IEEE International Conference on Image Processing, Lausanne, Sept. 1996, 129~132
- [25] Reusens E. Overlapped adaptive partitioning for image coding based on the theory of iterated function systems. In *Proceedings of ICASSP-1994 (IEEE International Conference on Acoustics, Speech and Signal Processing)*, Adelaide, 1994, 5: 569~572
- [26] Bone D J. Orthonormal fractal image encoding using overlapping blocks. *Fractals*, 1997, 5 (Supplementary issue): 187~199
- [27] Ho H, Cham W. Attractor image coding using lapped partitioned iterated function systems. In Proc. ICASSP 97, Munich, 1997, 4: 2917~2920
- [28] Forte B, Vrscay E. Theory of generalized fractal transform. In *Fractal Image Encoding and Analysis*, NATO ASI Series F, Fisher Y (Ed.), Berlin: Springer-Verlag, 1998
- [29] Cabrelli C A, Falsetti M C, Molter U M. Fractal Block-Coding: A Functional Approach for Image and Signal Processing. *Computers and Mathematics with Applications*, 2002, 44: 1183~1200
- [30] Tanimoto M, Katsuyama S, Ohyama H, Fujii T, Kimoto T. A new fractal image coding employing blocks of variable shapes. In Proc. ICIIP-96 IEEE International Conference on Image Processing, Lausanne, Sept. 1996, 1: 137~140
- [31] Chang Y, Shyu B, Wang S. Region-based fractal image compression with quadtree segmentation. In Proc ICASSP'97, Munich, 1997, 4: 3125~3128
- [32] Fisher Y (Ed.). *Fractal Image Compression: Theory and Application*. New York: Springer-Verlag, 1994
- [33] Saupe D, Hamzaoui R, Hartenstein H. Fractal image compression - An introductory overview. in *Fractal Models for Image Synthesis, Compression, and Analysis*, Saupe D, Hart J (eds.), ACM SIGGRAPH'96 Course Notes
- [34] Hurtgen B, Stiller C. Fast hierarchical codebook search for fractal coding of still images. in Proc. EOS/SPIE Visual Communications PACS Medical Applications, Berlin, Germany, 1993
- [35] Gharavi-Alkhansari M. Fractal-based image and video coding using matching pursuit. PhD Dissertation, University of Illinois, USA, 1997
- [36] Vitulano. Fractal image coding schemes using nonlinear grayscale functions. *Signal Processing (Elsevier)*, 2001, 81: 1095~1099
- [37] Lin H, Venetsanopoulos A N. Fractal-based image coding by nonlinear contractive functions. in: *Proceedings of the 17th Biennial Symposium on Communications*, Kingston, Ontario, May~June 1994, 295~298
- [38] Signes J. Geometrical interpretation of IFS based image coding. In *NATO ASI on Fractal Image Encoding and Analysis*, Trondheim, Norway, July 1995
- [39]   ien G E. Parameter quantization in fractal image coding. in Proc. IEEE Int. Conf. Image Processing, Austin, TX, Nov. 1994, III: 142~146
- [40] Jacquin A E. Fractal image coding: A review. *Proc. IEEE*, 1993, 81(10): 1451~1465

- [41] Hurtgen B, Mols P, Simon S F. Fractal transform coding of color images. in SPIE Proc.: Vis. Commun. Image Process., Katsaggelos A K(Ed.), July 1994, vol. 2308: 1683~1691
- [42] He C (何传江), Yang S X, Huang X (黄席樾). Progressive decoding method for fractal image compression. IEE Proc. -Vision, Image & Signal Processing, 2004, 151(3): 207~213
- [43] Barahav Z, Malah D, Karnin E. Hierarchical interpretation of fractal image coding and its application to fast decoding. in Proc. IEEE Int. Conf. Digital Signal Processing, Nicosia, Cyprus, July 1993, 190~195
- [44] Hamzaoui R. Decoding algorithm for fractal image compression. IEE Electronics Letters, 1996, 32 (14): 1273~1274
- [45] Kang H S, Kim S D. A fractal decoding algorithm for fast convergence. Opt. Eng., 1996, 35(11): 3191~3198
- [46] Hamzaoui R. Fast decoding algorithms for Fractal image compression. Technical Report 86, Institut für Informatik, Universität Freiburg, March 1997, <ftp://axes.informatik.uni-leipzig.de/pub/Fractal/papers>
- [47] Moon Y H, Kim H S, Kim J H. A Fast Fractal Decoding Algorithm Based on the Selection of an Initial Image. IEEE Transactions on Image Processing, 2000, 9(5): 941~945
- [48] Ruhl M, Hartenstein H. Optimal fractal coding is NP-hard. Proceedings DCC'97 Data Compression Conference, Storer J A, Cohn M (eds.), IEEE Computer Society Press, March 1997, 261~270
- [49] Lepsøy S, Øien G E, Ramstad T. Attractor image compression with a fast non-iterative decoding algorithm. Proceedings ICASSP'1993 IEEE International Conference on Acoustics, Speech and Signal Processing, 1993, 5: 337~340
- [50] Bedford T, Dekking F M, Breuer M, Keane M S, Schooneveld D. Fractal coding of monochrome images. Signal Processing: Image Communication, 1994, 6: 405~419
- [51] Fisher F, Jacobs E W, Boss R D. Fractal image compression using iterated transforms. in Image and Text Compression, Storer J A (Ed.), Kluwer Academic Publishers, Dordrecht, 1992, 35~61
- [52] Lee S, Omachi S, Aso H. VLSI architecture for quadtree-based fractal image coding. IEEE Proc.-Comput. Digit. Tech., 2001, 148(4/5)
- [53] Saupe D. The futility of square isometries in fractal image compression. In Proc. ICIP-96 IEEE International Conference on Image Processing, Lausanne, Sept. 1996
- [54] Ramamurthi B, Gersho A. Classified vector quantization of images. IEEE Trans. Commun., 1986, COM-34
- [55] Fisher Y. Fractal Image Compression. Fractals, 1994, 2 (3): 325~334
- [56] Caso G, Obrador P, Kuo C C J. Fast methods for fractal image encoding. in Proceedings from IS&T/SPIE 1995 Symposium on Electronic Imaging: Science & Technology, 1995, Vol. 2501: 583~594
- [57] Hurtgen B, Stiller C. Fast hierarchical codebook search for fractal coding of still images. in Proc. EOS/SPIE Visual Communications PACS Medical Applications, Berlin, Germany, 1993
- [58] Saupe D. From classification to multi-dimensional keys. in Fisher Y (ed.). Fractal image compression: Theory and Applications, New York: Springer-Verlag, 1994
- [59] Gersho A, Gray R. Vector Quantization and Signal Compression, Boston: Kluwer Academic Publishers, 1992
- [60] Tong C S, Wong M. Adaptive Approximate Nearest Neighbor Search for Fractal Image Compression. IEEE Transactions on Image Processing, 2002, 11(6): 605~615

- [61] Jeng J H, Truong T K, Sheu J R. Fast fractal image compression using the Hadamard transform. *IEE Proc.-Vis. Image Signal Process.*, 2000, 147(6)
- [62] Hartenstein H, Saupe D. Lossless acceleration of fractal image encoding via the fast Fourier transform, *Signal Processing: Image Communication*, 2000, 16: 383~394
- [63] 何传江, 李高平. 分形图像编码的改进算法. *计算机仿真*, 2004, 21(8): p.62~65
- [64] He C (何传江), Yang S X, Huang X (黄席樾). Variance-based accelerating scheme for fractal image encoding. *IEE Electronics Letters*, 2004, 40(2): 115~116
- [65] Lee C K, Lee W K. Fast Fractal Image Block Coding Based on Local Variances, *IEEE Transactions on Image Processing*, 1998, 7(6): 888~891
- [66] Lai C, Lam K, Siu W. Improved searching scheme for fractal image coding. *IEE Electronics Letters*, 2002, 38(25): 1653~2654
- [67] Davis G M. A wavelet-based analysis of fractal image compression. *IEEE Transactions on Image Processing*, 1998, 7(2): 141~154
- [68] Mallat S. A theory for multiresolution signal decomposition the wavelet representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1989, 11(7): 674~693
- [69] Barahav Z, Malah D, Karnin E. Hierarchical interpretation of fractal image coding and its application to fast decoding. in *Proc. IEEE Int. Conf. Digital Signal Processing*, Nicosia, Cyprus, July 1993, 190~195
- [70] Kima T, Van Dyck R E, Miller D J. Hybrid fractal zerotree wavelet image coding. *Signal Processing: Image Communication*, 2002, 17: 347~360
- [71] Barthel K U, Voyé T. Adaptive Fractal Image Coding in the Frequency Domain. *Journal on Communications*, 1994, XLV: 33~37
- [72] Barthel K U, Schuttemeyer J, Voyé T, Noll P. A new image coding technique unifying fractal and transform coding. *IEEE International Conference on Image Processing*, 1994, III: 112~116
- [73] Barthel K U. Entropy constrained fractal image coding. *Fractals*, 1997, 5(Suppl.): 17~26
- [74] Curtis K M, Neil G, Fotopoulos V. A hybrid fractal/DCT image compression method, *DSP2002*, 0-7803-7503-3/02, ©2002 IEEE, 1337~1340
- [75] Skodras A N. Fast Discrete Cosine Transform Pruning. *IEEE Trans. on Signal Processing*, 1994, 42(7): 1833~1837
- [76] Zhao Y, Yuan B. Image compression using fractals and discrete cosine transform, *IEE Electronics Letters*, 1994, 30(6): 474~475
- [77] Wohlberg B, De Jager G. Fast image domain fractal compression by DCT domain block matching, *IEE Electronics Letters*, 1995, 31(10): 869~870
- [78] Au O C, Liou M L, Ma L K. Fast fractal encoding in frequency domain. in *Proceedings of IEEE International Conference on Image Processing ICIP'97*, Santa Barbara, October 1997, 298~301
- [79] Yuxuan R, Nge T G.. An improved fractal image compression scheme embedding DCT encoder. *Image Processing and its Applications*, Conference Publication No. 465 © IEE 1999, 610~614
- [80] Truong T, Jeng J, Reed I S, Lee P C, Li A Q. A fast encoding algorithm for fractal image compression using the DCT inner product. *IEEE Transactions on Image Process*, 2000, 9(4): 529~535
- [81] Mas Ribes J M, Simon B, Macq B. Combined Kohonen neural networks and discrete cosine transform method for iterated transformation theory. *Signal Processing: Image Communication*, 2001, 16: 643~656
- [82] Melnikov G, Katsaggelos A K. A jointly optimal fractal/DCT compression scheme. *IEEE transactions on multimedia*, 2002, 4(4): 413~422

- [83] Farhadi G. A hybrid image compression scheme using block-based fractal coding and DCT. EC-VIP-MC 2003, 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications, 2-5 July 2003, Zagreb, Croatia, 89~94
- [84] Jeng J H, Truong T K, Sheu J R. Fast fractal image compression using the Hadamard transform. IEE Proc.-Vis. Image Signal Process., 2000, 147(6)
- [85] Kim K, Park R. Still image coding based on vector quantization and fractal approximation. IEEE Transactions on Image Processing, 1996, 5(4): 587~597
- [86] Kim C, Kim R, Lee S. A fractal vector quantizer for image coding. IEEE Transactions on Image Processing, 1998, 7(11): 1598~1602
- [87] Hamzaoui R, Saupe D. Combining fractal image compression and vector quantization, IEEE Transactions on Image Processing, 2000, 9(2): 197~208
- [88] Nappi M, Vitulano D. Linear prediction image coding using iterated function systems, Image and Vision Computing, 1999, 17: 771~776
- [89] Tong C S, Pi M. Analysis of a hybrid fractal-predictive-coding compression scheme, Signal Processing: Image Communication, 2003, 18: 483~495
- [90] Wang Z, Zhang D, Yu Y. Hybrid image coding based on partial fractal mapping, Signal Processing: Image Communication, 2000, 15: 767~779
- [91] Sun K T, Lee S J, Wu P Y. A Neural network approaches to fractal image compression and decompression. Neurocomputing, 2001, 41: 91~107
- [92] Mitra S K, Murthy C A, Kundu M K. Technique for fractal image compression using genetic algorithm, IEEE Transactions on Image Processing, 1998, 7(4): 586~593
- [93] He C (何传江), Yang S X, Xu X. Fast fractal image compression based on one-norm of normalised block, IEE Electronics Letters, 2004, 40(17): 1052~1053
- [94] Huang X (黄席隼), He C (何传江), Yang S X, Ma L. Partitioned iterated function systems with Lipschitz grayscale maps for fractal image coding. Dynamics Of Continuous Discrete and Impulsive Systems-Series Bapplications & Algorithms, 2003, Suppl. S: 225~230
- [95] He C (何传江), Huang X (黄席隼), Jiang H, Liu T. A novel solution to reduce tile effect exhibited by fractal image compression techniques, Proceedings of the International Conference on Wavelet Analysis and Its Applications (WAA), 2003, 1: 7~12
- [96] 何传江, 蒋海军, 黄席隼. 快速分形图像编码局部方差算法的改进. 计算机仿真, 2004, 21(6): 141~144
- [97] 何传江, 蒋海军, 黄席隼. 基于平均偏差排序的分形图像快速编码. 中国图像图形学报, 2004, 9(9): 1130~1134
- [98] 何传江, 蒋海军, 黄席隼. 快速分形图像编码的一种特征方法. 电子学报, 2004, 32(11): 1864~1867

第三部分 蚁群优化算法 理论及其应用



第九章 蚁群优化算法概述

第一节 引言

人工智能在经历了 20 世纪 80 年代整整 10 年的繁荣后, 由于方法论上始终没有突破经典计算思想的藩篱, 再次面临着寒冬季节的考验^[1]. 以 Penrose 等^[2,3]为代表的各种反思和批驳论著纷纷涌现, 人工智能的研究前景又一次变得暗淡无光. 与此同时, 随着人们对生命本质的不断了解, 生命科学却以前所未有的速度迅猛发展, 使人工智能的研究开始摆脱经典逻辑计算的束缚, 大胆探索起新的非经典计算途径. 正如人工智能先驱 Minsky 所认为的“我们应该从生物学而不是物理学受到启示…….”^[4]那样, 对生物启发式计算 (Bio-inspired Computing) 的研究, 成为人工智能迎接新曙光而开启的又一个春天.

在这种背景下, 社会性动物 (如蚁群、蜂群、鸟群等) 的自组织 (Self-organization) 行为引起了人们的广泛关注, 许多学者对这种行为进行数学建模并用计算机对其进行仿真, 这就产生了所谓的“群集智能” (Swarm Intelligence, 简称 SI)^[5~10]. 社会性动物的妙处在于: 个体的行为都很简单, 但当他们一起协同工作时, 却能够“突现” (Emerge) 出非常复杂 (智能) 的行为特征. 例如, 单只蚂蚁的能力极其有限, 但当这些简单的蚂蚁组成蚁群时, 却能完成像筑巢、觅食、迁徙、清扫蚁巢等复杂行为; 一群行为显得盲目的蜂群能造出精美的蜂窝; 鸟群在没有集中控制的情况下能够同步飞行等.

在这些自组织行为中, 又以蚁群在觅食过程中总能找到一条从蚁巢到食物源的最短路径最为引人注目. 受其启发, 意大利学者 M Dorigo, V Maniezzo 和 A Colorni^[11~13] 于 20 世纪 90 年代初提出了一种新型的智能优化算法——蚂蚁系统 (ant system, 简称 AS), 该算法首先用于求解著名的旅行商问题 (traveling salesman Problem, 简称 TSP) 并获得了较好的效果. 在 20 世纪 90 年代中期, 这种算法逐渐引起了许多研究者的注意, 并对其作了各种改进或将其应用于更为广泛的领域, 取得了一些令人鼓舞的成果^[6~8,10,12]. 近几年, M Dorigo 等人^[6,14] 将蚂蚁算法进一步发展成一种通用的优化技术——蚁群优化 (ant colony optimization, 简称 ACO), 并将所有符合 ACO 框架的蚂蚁算法称为蚁群优化算法 (ACO algorithm), 从而为 ACO 的理论研究和算法设计提供了一个统一的框架. W J Gutjahr^[15] 首先对 ACO 的收敛性进行了探讨, 取得了一些初步的结果, 目前已有少量文献涉及 ACO 的收敛性^[16~19], 但还很不成熟. 对 ACO 的应用研究一直非常活跃, 目前该算法已在求解组合优化^[14,20~44]、函数优化^[45~48]、系统辨识^[49]、机器人路径

规划^[50~52]、数据挖掘^[53]、网络路由^[54~65]等问题时取得了很好的效果。

目前, ACO 在启发式方法范畴内已逐渐成为一个独立的分支, 在有关国际会议上多次作为专题加以讨论, 如 1998 年、2000 年、2002 年在比利时布鲁塞尔大学召开了三届 ACO 国际研讨会 (Ants'98, Ants'2000, Ants'2002), 第四届 ACO 国际研讨会将于 2004 年 9 月在同一地点举行, 届时将就 SI 算法, ACO 算法和群集机器人技术 (swarm robotics, 简称 SR) 进行讨论。

尽管目前对 ACO 的研究刚刚起步, 一些思想尚处于萌芽时期, 但人们已隐隐约约认识到, 人类诞生于大自然, 解决问题的灵感似乎也应该来自大自然, 因此, 越来越多人开始关注和研究 ACO, 初步的研究结果已显示出该算法在求解复杂优化问题 (特别是离散优化问题) 方面的优越性。虽然 ACO 的严格理论基础尚未奠定, 国内外的有关研究仍停留在实验探索阶段, 但从当前的应用效果来看, 这种模仿自然生物的新型系统寻优思想无疑具有十分光明的前景。

第二节 蚁群优化原理及算法描述

一、蚁群的自组织行为

自然界中蚁群的自组织行为很早就引起了昆虫学家的注意^[67~69]。Deneubourg^[70]等通过“双桥实验”对蚁群的觅食行为进行了研究。如图 9-1(a) 所示, 对称双桥 (两座桥的长度相同) A、B 将蚁巢与食物源分隔开, 蚂蚁从蚁巢自由地向食物源移动。图 9-1(b) 是经过 A、B 两桥的蚂蚁百分比随时间的变化情况。实验结果显示, 在初始阶段出现一段时间的震荡 (由于某些随机因素, 使通过某座桥上的蚂蚁数急剧增多或减少) 后, 蚂蚁趋向于走同一条路径。在该实验中, 绝大部分蚂蚁选择了 A 桥。

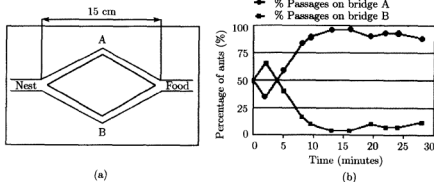


图 9-1 对称双桥实验 (Deneubourg et al., 1990)

在实验初期, A、B 两座桥上都没有外激素^①存在, 蚂蚁将以相同的概率选择 A、B 两座桥, 故此时蚂蚁在两座桥上留下的外激素量相等。经过一段时间后, 由于随机波动致使大部分蚂蚁选择了 A 桥 (也有可能为 B 桥), 因此更多的外激素将会留在 A 桥上, 致使 A 桥对后来的蚂蚁有更大的吸引力。如图 9-1(b) 所示, 随着时间的推移, A 桥上的蚂蚁数将越来越多, 而 B 桥上正好相反。

S Goss^[71] 等人给出了上述实验的概率模型。首先, 假定桥上残留的外激素量与过去一段时间经过该桥的蚂蚁数成正比 (这意味着不考虑外激素蒸发的情况); 其次, 某一时刻蚂蚁按桥上外激素量的多少来选择某座桥, 即蚂蚁选择某座桥的概率与经过该桥的蚂蚁数成正比。当所有 m 只蚂蚁都经过两座桥以后, 设 A_m 、 B_m 分别为经过 A 桥和 B 桥的蚂蚁数 ($A_m + B_m = m$), 则第 $m+1$ 只蚂蚁选择 A 桥的概率为

$$P_A(m) = \frac{(A_m + k)^h}{(A_m + k)^h + (B_m + k)^h}, \quad (9.2.1)$$

而选择 B 桥的概率为

$$P_B(m) = 1 - P_A(m), \quad (9.2.2)$$

其中参数 h 和 k 用来匹配真实实验数据。第 $(m+1)$ 只蚂蚁首先按式 (9.2.1) 计算选择概率 $P_A(m)$, 然后生成一个在区间 $[0, 1]$ 上一致分布的随机数 φ , 如果 $\varphi \leq P_A(m)$, 则选择 A 桥, 否则选择 B 桥。为了求得参数 k 和 h , 通过蒙特卡罗模拟证实^[72], 当 $k \approx 20$, $h \approx 2$ 时, 式 (9.2.1) 与实验数据相一致。

另外, Goss^[71] 等人还用非对称双桥 (两座桥的长度不相等) 进行了实验。如图 9-2 所示, 图 9-2(a) 为蚂蚁经过非对称双桥开始觅食; 图 9-2(b) 显示绝大多数

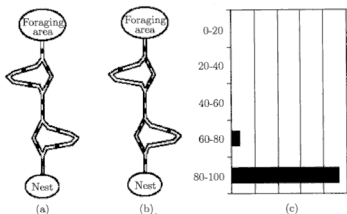


图 9-2 非对称双桥实验 (Goss et al., 1989)

① 蚂蚁在其经过的路径上留下的一种物质, 后来的蚂蚁能够感知到这种物质, 且倾向于朝该物质浓度强的方向移动。

蚂蚁选择较短的桥；图 9-2(c) 显示最终有 80%~100% 的蚂蚁选择较短的桥。

在非对称双桥实验中，随机抖动对胜出桥（有较多蚂蚁选择的桥）的影响减小，而占主导作用的是随机外激素的引导行为。

除能够找到蚁巢和食物源之间的最短路径外，蚁群还有极强的适应环境的能力，如图 9-3 所示，在蚁群经过的路线上突然出现障碍物时，蚁群能够很快重新找到新的最优路径。

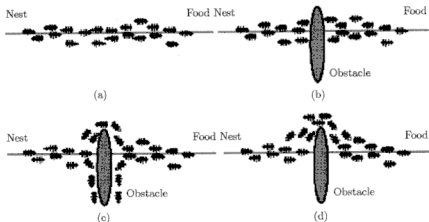


图 9-3 蚁群的自适应行为 (M Dorigo et al., 1996)

- (a) 蚁群在蚁巢和食物源之间的路径上移动；(b) 路径上出现障碍物；
(c) 较短路径上的外激素以更快的速度增加；(d) 所有的蚂蚁都选择较短的路径。

二、蚁群优化的原理分析

ACO 是受自然界中真实蚁群的集体觅食行为的启发而发展起来的一种基于群体的模拟进化算法，属于随机搜索算法。M Dorigo 等人充分利用了蚁群搜索食物的过程与著名 TSP 之间的相似性，通过人工模拟蚁群搜索食物的行为来求解 TSP。

从上节的“双桥实验”可以看出，像蚂蚁这类社会性动物，虽然个体的行为极其简单，但由这些简单个体所组成的蚁群却表现出极其复杂的行为特征。如蚁群除了能够找到蚁巢与食物源之间的最短路径外，还能适应环境的变化，即在蚁群运动的路线上突然出现障碍物时，蚂蚁能够很快地重新找到最短路径。蚁群是如何完成这些复杂任务的呢？仿生学家经过大量的观察、研究发现，蚂蚁在寻找食物时，能在其经过的路径上释放一种蚂蚁特有的分泌物——外激素 (Pheromone)，使得一定范围内的其他蚂蚁能够感觉到这种物质，且倾向于朝着该物质强度高的方向移动。因此，蚁群的集体行为表现为一种信息正反馈现象：某条路径上经过的蚂蚁数越多，其上留下的外激素的迹也就越多（当然，随时间的推移会逐渐蒸发掉一部分），后来蚂蚁选择该路径的概率也越高，从而更增加了该路径上外激素的强度。蚁群这种选择路径的过程被称之为自催化行为 (autocatalytic behavior)，由于

其原理是一种正反馈机制,因此也可将蚁群的行为理解成所谓的增强型学习系统(reinforcement learning system)^[73,74].

下面用图 9-4 解释蚁群发现最短路径的原理和机制.

如图 9-4(a) 所示,在蚁巢和食物源之间有两条道路 Nest-A-B-D-Food 和 Nest-A-C-D-Food,其长度分别为 4 和 6.单位时间内蚂蚁可移动一个单位长度的距离.开始时所有路径上都没有外激素.

如图 9-4(b),在 $t=0$ 时刻,20 只蚂蚁从蚁巢出发移动到 A.由于路径上没有外激素,他们以相同概率选择左侧或右侧道路,因此平均有 10 只蚂蚁走左侧,另外 10 只走右侧.

如图 9-4(c),在 $t=4$ 时刻,第一组先到达食物源的蚂蚁将折回.

如图 9-4(d),在 $t=5$ 时刻,两组蚂蚁将在 D 点相遇.此时 BD 上的外激素数量与 CD 上的相同,因此返回的 10 只蚂蚁中有 5 只选择 BD 而另 5 只选择 CD.

如图 9-4(e),在 $t=8$ 时刻,前 5 个蚂蚁将返回巢穴,而在 AC、CD 和 AB 上各有 5 个蚂蚁.

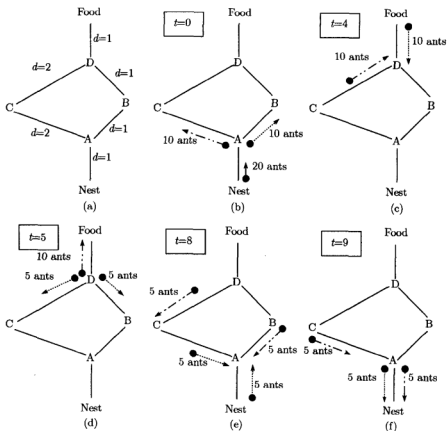


图 9-4 蚁群寻找最短路径的原理

如图 9-4(f), 在 $t=9$ 时刻, 前 5 个蚂蚁又回到 A 并且再次面对往左还是往右的选择。这时, AB 上的轨迹数是 20 而 AC 上是 15, 因此将有较为多数的蚂蚁选择往右, 从而增强了 AB 上外激素的量。随着该过程的继续, 两条道路上外激素数量的差距将越来越大, 直至绝大多数蚂蚁都选择了最短的路径。正是由于一条道路要比另一条道路短, 因此, 在相同的时间间隔内, 短的路线会有更多的机会被选择。

三、蚁群优化算法描述

ACO 算法可以看成是一种基于解空间参数化概率分布模型 (Parameterized Probabilistic Model) 的搜索算法框架^[75~77], 其中解空间参数化概率模型的参数就是信息素, 因而这种模型就是信息素模型。在基于模型的搜索算法框架中, 可行解通过在一个解空间参数化概率分布模型上的搜索产生, 此模型的参数用以前产生的解来进行更新, 使得在新模型上的搜索能集中在高质量的解搜索空间内。这种方法的有效性建立在高质量的解总是包含好的解构成元素的假设前提下。通过学习这些解构成元素对解的质量的影响有助于找到一种机制, 并通过解构成元素的最佳组合来构造出高质量的解。一般来说, 一个基于模型的搜索算法通常使用以下两步迭代来解决优化问题:

(1) 可行解通过在解空间参数化概率分布模型上的搜索产生;

(2) 用搜索产生的解来更新参数化概率模型, 即更新解空间参数化概率分布的参数, 使得在新模型上的搜索能集中在高质量的解搜索空间内。

在蚁群算法中, 基于信息素的解空间参数化概率模型 (信息素模型) 以解构造图的形式给出。在解构造图上, 定义了一种作为随机搜索机制的人工蚂蚁, 蚂蚁通过一种分布在解构造图上被称为信息素的局部信息的指引, 在解构造图上移动, 从而逐步地构造出问题的可行解。信息素与解构造图上的节点或弧相关联, 作为解空间参数化概率分布模型的参数。

由于 TSP 能够非常直接地映射为解构造图 (城市为节点, 城市间的路径为弧, 信息素分布在弧上), 且是一个 NP 难题, 因而蚁群算法的大部分成果都集中在 TSP 上。一般而言, 用于求解 TSP、生产调度问题等组合优化问题的蚁群算法都遵循如下的统一算法框架。

求解组合优化问题的蚁群算法 (ACO-COP) 为:

algorithm ACO-COP

 设置参数, 初始化信息素踪迹

 while (不满足结束条件)do

 for 蚁群中的每只蚂蚁

 for 每个解构造步 (直到构造出完整的可行解)

 1) 蚂蚁按信息素及启发式信息的指引构造一步问题的解;

```
2) 进行信息素局部更新. (可选)
end for
end for
1) 以某些已获得的解为起点进行邻域 (局部) 搜索; (可选)
2) 根据某些已获得的解的质量进行全局信息素更新.
end while
end
```

在该算法中, 蚂蚁逐步地构造问题的可行解, 在一步解的构造过程中, 蚂蚁以概率方式选择信息素强且启发式因子高的弧到达下一个节点, 直到不能继续移动为止. 此时, 蚂蚁走过的路径对应待求解问题的一个可行解. 局部信息素更新针对蚂蚁当前走过的一步路径上的信息素进行, 全局信息素更新是在所有蚂蚁找到可行解之后, 根据发现解的质量或当前算法找到的最好解对路径上的信息素进行更新.

第三节 蚁群优化的特点

从 ACO 的原理不难看出, 蚁群的觅食行为实际上是一种分布式的协同优化机制. 单只蚂蚁虽然能够找到从蚁巢到食物源的一条路径, 但找到最短路径的可能性极小, 只有当多只蚂蚁组成蚁群时, 其集体行为才展现出蚂蚁的智能——发现最短路径的能力. 在寻找最短路径的过程中, 蚁群使用了一种间接的通信方式, 即通过向所经过的路径上释放一定量的外激素, 其他蚂蚁通过感知这种物质的强弱来选择下一步要走的路. 这种个体间通过改变环境、感知环境的变化来彼此间接通讯的方式机制被称为协同机制 Stigmergy^[78]. 该通信机制可以非常容易地扩展到人工多主体模型 (Artificial Multi-agent Model, 简称 AMM). 即首先用状态变量来表示问题的状态, 然后让人工主体只访问局部状态变量信息, 如蚂蚁通过感知前面蚁群留下的外激素来完成觅食工作. 因此, 人工蚂蚁可以通过更新问题的状态变量来模拟真实蚂蚁更新外激素的行为.

在蚁群的觅食行为中, 另一个重要的方面是自催化机制和解的隐式评估. 自催化机制实际上是一种正反馈机制, 解的隐式评估指蚁群将先走完较短的路径. 自催化机制和解的隐式评估相结合, 极大地提高了问题的求解效率. 即对于越短的路径, 蚂蚁将越早走完, 从而使更多的蚂蚁将会选择该路径. 自催化机制对基于群体的算法非常有效, 如在遗传算法 (Genetic Algorithm, 简称 GA) 中, 通过选择和复制机制来实现. 因为他奖励好的个体, 可以指导搜索方向. 当然在使用自催化机制时, 要努力避免早熟现象. 在 ACO 中, 使用外激素蒸发和随机状态转移来弥补自催化机制的缺陷.

ACO 的主要特点概括如下:

- (1) 采用分布式控制, 不存在中心控制;
- (2) 每个个体只能感知局部的信息, 不能直接使用全局信息;
- (3) 个体可改变环境, 并通过环境来进行间接通讯 (Stigmergy 机制);
- (4) 具有自组织性, 即群体的复杂行为是通过个体的交互过程中突现出来的智能 (Emergent Intelligence);
- (5) 是一类概率型的全局搜索方法, 这种非确定性使算法能够有更多的机会求得全局最优解;
- (6) 其优化过程不依赖于优化问题本身的严格数学性质, 诸如连续性、可导性及目标函数和约束函数的精确数学描述;
- (7) 是一类基于多主体 (Multi Agent) 的智能算法, 各主体间通过相互协作来更好地适应环境;
- (8) 具有潜在的并行性, 其搜索过程不是从一点出发, 而是同时从多个点同时进行. 这种分布式多智能体的协作过程是异步并发进行的, 分布并行模式将大大提高整个算法的运行效率和快速反应能力.

第四节 蚁群优化与其他算法的关系

一、蚁群优化与启发式图搜索

在 ACO 中, 每只蚂蚁在问题的解空间中按启发式图进行搜索. 蚂蚁按概率决策准则选择下一个要到的节点, 其中, 概率决策准则使用启发式评估函数来指导蚂蚁选择更有希望的节点.

二、蚁群优化与蒙特卡罗模拟

可以将 ACO 解释为并行的重复蒙特卡罗 (Monte Carlo, 简称 MC) 系统. 蒙特卡罗系统是通用的随机模拟系统, 即通过利用随机状态采样和转移准则, 对问题进行重复的采样实验^[79]. 实验所得的结果对问题的统计知识和感兴趣变量的估计值进行更新. 反过来, 可以重复地使用知识以减少感兴趣变量的不一致性, 从而指导模拟过程向感兴趣的状态空间转移. 与此相似, 在 ACO 中, 蚂蚁利用随机决策机制在问题的解空间中逐步发现问题的可行解. 每只蚂蚁自适应地修改问题的局部信息 (即蚁群留下的外激素), 通过 Stigmergy 机制指导蚂蚁沿着有希望的解空间向最优解靠近, 从而节约了算法的搜索时间.

三、蚁群优化与神经网络

由许多并发、局部交互的单元 (人工蚂蚁) 组成的蚁群, 可以看成是一种“连接”

系统,“连接”系统最具代表性的例子是神经网络(neural network,简称 NN)^[80,81]。从结构上看,ACO 与通常的神经网络具有类似的并行机制。蚂蚁访问的每一个状态 i 对应于神经网络中的神经元 i ,与问题相关的状态 i 的邻域结构与神经元 i 中的突触连接相对应。蚂蚁本身可看成通过神经网络的并发输入信号,以修改突触与神经元之间的连接强度。信号经过随机转换函数的局部反传,使用的突触越多,两个神经元之间的连接越强。ACO 中的学习规则可解释为一种后天性的规则,即质量较好的解包含连接信号的强度高于质量较差的解。

四、蚁群优化与进化计算

ACO 与进化计算(evolutionary computation,简称 EC)^[82]之间有许多相似之处。首先,两种算法都采用群体表示问题的解;其次,新群体通过包含在群体中与问题相关的知识来生成。两者的主要差异在于进化计算中所有问题的知识都包含在当前群体中,而 ACO 中代表过去所学的知识保存在信息素的迹中。

五、蚁群优化与随机学习自动机

随机学习自动机(stochastic learning automata,简称 SLA)^[83]是最古老的机器学习方法之一。自动机可定义为一组可能的操作和一个相关的概率向量,一组连续的输入和一个学习算法用来学习输入-输出之间的关系。自动机与一个正反馈的环境相关联,同时还定义了一组环境对行为的惩罚信号。SLA 与 ACO 的相似性为:在 ACO 中,每条弧上信息素的迹可看成并发的随机学习过程,蚂蚁扮演环境信号的角色,信息素的更新规则相当于自动机的学习规则。两者的主要差别在于 ACO 中的环境信号是一种随机的、通过概率转移规则的偏差,学习过程沿着搜索空间中最感兴趣的部分进行,即整个环境扮演了一个关键的角色,以此来学习好的解空间。

第五节 蚁群优化的研究现状

1991 年, M Dorigo 等提出了第一个 ACO 算法——蚂蚁系统(AS)并成功用于求解 TSP^[10~13]。实验结果表明 AS 算法具有较强的鲁棒性和发现较好解的能力,但同时也存在一些缺陷,如收敛速度慢、易出现停滞现象等。该算法的出现引起了学者们的广泛关注,并提出了一些改进的 ACO 算法。L M Gambardella, M Dorigo^[74]提出了 Ant-Q 算法,该算法用伪随机比例状态转移规则(Pseudo random proportional state transition rule)替换 AS 算法中的随机比例选择规则(Stochastic proportional choice rule),从而使 Ant-Q 算法在构造解的过程中能够更好地保持知

识探索 (Exploration) 与知识利用 (Exploitation) 之间的平衡。除此之外, 该算法中还引入了局部信息素更新机制和全局信息素更新中的精英策略。M Dorigo^[84] 等在 Ant-Q 算法的基础上提出了蚁群系统 (ant colony system, 简称 ACS), 该算法作为 Ant-Q 算法的特例实现起来更为简单, 但在求解 TSP 时具有相同的性能。Stützle 和 Hoos^[85~88] 提出了最大-最小蚂蚁系统 (max-min ant system, 简称 MMAS), 该算法的主要特点是为信息素设置上下限来避免算法出现停滞现象。Bullnheimer 等^[89] 提出了基于排序的蚂蚁系统 (rank-based version of ant system, AS_{rank}), 该算法在完成一次迭代后, 将蚂蚁所经路径的长度按从小到大的顺序排列, 并根据路径长度赋予不同的权重, 路径较短的权重较大, 如全局最优解的权重为 w , 第 r 个最优解的权重为 $\max\{0, w - r\}$ 。

国内直到 20 世纪末才有学者开始关注 ACO 算法, 目前对该算法的研究主要停留在算法的改进和应用方面。吴庆洪和张纪会等^[90] 通过向基本蚁群算法中引入变异机制, 充分利用 2-交换法简洁高效的特点, 提出了具有变异特征的蚁群算法。吴斌和史忠植^[26] 首先在蚁群算法的基础上提出了相遇算法, 提高了蚂蚁一次周游的质量, 然后将相遇算法与采用并行策略的分段算法相结合, 提出一种基于蚁群算法的 TSP 分段求解算法。王颖和谢剑英^[91] 通过自适应地改变算法的挥发度等系数, 提出一种自适应的蚁群算法以克服限于局部最小的缺点。覃刚力和杨家本^[92] 根据人工蚂蚁所获得解的情况, 动态地调整路径上的信息素, 提出了自适应调整信息素的蚁群算法。

随着人们对 ACO 研究的不断深入, 近年来 M Dorigo 等人^[6,14] 提出了蚁群优化元启发式 (ant colony optimization meta heuristic, 简称 ACO-MH) 这一求解复杂问题的通用框架。ACO-MH 为 ACO 的理论研究和算法设计提供了技术上的保障。在 ACO 的收敛性方面, W J Gutjahr^[15~17] 作了开创性的工作, 提出了基于图的蚂蚁系统元启发式 (graph-based ant system metaheuristic) 这一 ACO 的通用模型, 该模型在一定的条件下能以任意接近 1 的概率收敛到最优解。文献^[17] 证明了与模拟退火 (simulated annealing, 简称 SA) 相似的结果, 即算法能以概率 1 收敛到最优解。T Stützle 和 M Dorigo^[18] 对一类 ACO 算法的收敛性进行了证明, 其结论可以直接用到两类实验上证明是最成功的 ACO 算法——MMAS 和 ACS。N Meuleau 和 M Dorigo^[19] 研究了随机梯度下降 (stochastic gradient descent, 简称 SGD) 和 ACO 之间的关系, 将 ACO 看成是一种近似的 SGD 算法, 并根据 SGD 实现了理论上收敛的 ACO 算法。

对 ACO 的应用研究一直非常活跃。继 M Dorigo 首先将 AS 算法应用于 TSP 之后, V Maniezzo^[29,30] 等人将 AS 算法应用于指派问题 (quadratic assignment problem, 简称 QAP)。最近几年, Gambardella^[28,32], Taillard^[31] 和 Stützle^[88] 也发表了一些用 ACO 算法求解 QAP 的文章。目前, ACO 已是求解 QAP 最有效的算法之一。

A Colomi 等人^[37] 首先将 AS 算法应用于车间作业调度问题 (job-shop schedul-

ing problem, 简称 JSP). 实验结果表明, 该算法虽然能解决 JSP, 但同 state-of-the-art 相比较, 并没有任何优势. 但在用 MMAS 算法求解流动车间调度问题 (flow shop scheduling problem, 简称 FSSP) 时, 实验结果显示 MMAS 算法优于模拟退火和禁忌搜索算法.

Costa 和 Herz^[40] 提出增强的 AS 算法, 并将其应用于分配类型的问题. 该算法在求解图的着色问题时, 得到了完全可以和其他启发式算法相媲美的结果.

Bullnheimer, Hartl 和 Strauss^[41~43] 用算法 AS_{rank} 来求解车辆路径问题 (vehicle routing problems, 简称 VRP), 取得了和最优解非常相近的结果. 近几年, Gambardella, Taillard 和 Agazzi^[44] 对 VRP 问题的研究, 对一些基准问题 (benchmark problem) 的最优解有所提高.

ACO 在通讯网络领域 (特别是解决网络路由问题) 的应用受到越来越多学者的关注^[54~65]. 由于网络中信息的分布式性、动态性、随机性和异步性与 ACO 非常相似, 如利用局部信息发现解, 间接的通讯方式和随机状态的转换. Di Caro 和 Dorigo^[56,57] 已在相关的文献中将 ACO 应用于网络路由问题, 并称这种算法为 AntNet.

除了各种组合优化问题之外, ACO 算法还在函数优化、系统辨识、机器人路径规划、数据挖掘、大规模集成电路中的综合布线设计等领域取得了引人注目的成果^[20~66,84~112], 表 9-1 列举了有代表性的 ACO 算法及其应用情况.

表 9-1 蚁群优化算法及其应用

问题名称	研究者	算法名称	年代
旅行商问题 (Traveling salesman problem, TSP)	Dorigo, Maniezzo & Colorni	AS	1991
	Gambardella & Dorigo	Ant-Q	1995
	Dorigo & Gambardella	ACS & ACS-3-opt	1996
	Stützle & Hoos	MMAS	1997
	Bullnheimer, Hart & Strauss	AS _{rank}	1997
	Cordón, et al.	BWAS	2000
指派问题 (Quadratic assignment problem, QAP)	Maniezzo, Colorni & Dorigo	AS-QAP	1994
	Gambardella, Taillard & Dorigo	HAS-QAP	1997
	Stützle & Hoos	MMAS-QAP	1997
	Maniezzo	ANTS-QAP	1998
	Maniezzo	AS-QAP	1999
调度问题 (Scheduling problem)	Colorni, Dorigo & Maniezzo	AS-JSP	1994
	Stützle	AS-FSP	1997
	Bauer et al.	ACS-SMTTP	1999
	Den Besten, Stützle & Dorigo	ACS-SMTWTP	1999
	Merkle, Middendorf & Schmeck	ACO-RCPS	2000
	Li Yan-jun & Wu Tie-jun	NACA	2003
车辆路径问题 (Vehicle routing problem, VRP)	Bullnheimer, Hartl & Strauss	AS-VRP	1997
	Gambardella, Taillard & Agazzi	HAS-VRP	1999

续表

问题名称	研究者	算法名称	年代
图着色问题	Costa & Hertz	ANTCOL	1997
面向非连接的网络路由 (Connection-less network routing)	Di Caro & Dorigo Subramanian, Druschel & Chen Heusse et al. Van der Put & Rothkrantz	AntNet&AntNet-FA Regular ants CAF ABC-backward	1997 1997 1998 1998
面向连接的网络路由 (Connection-less network routing)	Schoonderwoerd et al. White, Pagurek & Oppacker Di Caro & Dorigo Bonabeau et al.	ABC ASGA AntNet-FS ABC-smart ants	1996 1998 1998 1998
Sequential ordering	Gambardella & Dorigo	HAS-SOP	1997
Shortest common supersequence	Michel & Middendorf	AS-SCS	1998
Frequency assignment	Maniezzo & Carbonaro	ANTS-FAP	1998
Generalized assignment	Ramalhinho Lourenco & Serra	MMAS-GAP	1998
多背包问题 (Multiple knapsack problem, MKP)	Leguizamón & Michalewicz	AS-KP	1999
Optical networks routing	Navarro Varela & Sinclair	ACO-VWP	1999
Redundancy allocation	Liang & Smith	ACO-RAP	1999
Constraint satisfaction	Solnon	Ant-P-solver	2000
机器人路径规划问题	金飞虎, 洪炳熔等	ACA	2002
数据挖掘 (Data mining)	Rafael S Parpinelli, Heitor S Lops, et al.	Ant-Miner	2002
连续函数优化	汪镭, 吴启迪	ACA	2003
系统辨识	汪镭, 吴启迪	AS	2003

第六节 本章小结

蚁群优化算法是从自然界中蚂蚁的觅食行为受到启发而提出的一种模拟进化算法, 初步的研究结果已显示出该算法的有效性。本章首先分析了蚂蚁的自组织行为, 并由此阐述了蚁群优化算法的基本原理及算法描述。阐述了蚁群优化算法的特点, 同时将其与其他算法进行了对比。最后, 对蚁群优化的国内外研究现状进行了综述。

参 考 文 献

- [1] 周昌乐. 心脑计算举要. 北京: 清华大学出版社, 2003. 237
- [2] Penrose R. The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics. New York: Oxford University Press, 1989. 640
- [3] Penrose R. Shadows of the Minds, A Search for the Missing Science of Consciousness. New York: Oxford University Press, 1994. 480
- [4] Minsky M. Logical Versus Analogical or Symbolic Versus Connectionist or Neat Versus Scruffy. AI Magazine, 1991, 12(2): 34~51

- [5] Colorni A, Dorigo M, Maffioli F, Maniezzo V, Righini G, and Trubian M. Heuristics from nature for hard combinatorial problems. *International Transactions in Operational Research*, 1996, 3(1): 1~21
- [6] Dorigo M and Di Caro G. The Ant Colony Optimization meta-heuristic. In Corne D, Dorigo M, and Glover F, *New Ideas in Optimization*. London: McGraw Hill, 1999. 11~32
- [7] Bonabeau E, Dorigo M, and Theraulaz G. *Swarm intelligence: from natural to artificial systems*. New York: Oxford University Press, 1999
- [8] Kennedy J, Eberhart R C, and Yuhui Shi. *Swarm Intelligence*. San Francisco: Morgan Kaufmann, 2001. 512
- [9] Steven Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York Simon & Schuster Adult Publishing Group, 2002. 288
- [10] Dorigo M and Stützle T. *Ant Colony Optimization*. MIT Press, 2004. 328
- [11] Dorigo M, Maniezzo V, and Colorni A. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991
- [12] Dorigo M. *Optimization, Learning and Natural Algorithms*(in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, IT, 1992
- [13] Colorni A, Dorigo M, and Maniezzo V. Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*. Elsevier, 1992. 134~142
- [14] Dorigo M, Di Caro G, and Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, 5(2): 137~172
- [15] Gutjahr W J. A generalized convergence result for the graph-based ant system metaheuristic. Dept. Statistics and Decision Support Systems, Univ. Vienna, Austria, Tech. Rep. 99-09, 1999
- [16] Gutjahr W J. A graph-based ant system and its convergence. *Future Gener. Comput. Syst.*, 2000, 16(8): 873~888
- [17] Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. *Info. Processing Lett.*, 2002, 82(3): 145~153
- [18] Stützle T and Dorigo M. A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(4): 358~365
- [19] Meuleau N and Dorigo M. Ant colony optimization and stochastic gradient descent. *Artif. Life*, 2002, 8(2): 103~121
- [20] 马良, 项培军. 蚂蚁算法在组合优化中的应用. *管理科学学报*, 2001, 4(2): 32~37
- [21] Dorigo M, Maniezzo V, and Colorni A. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 1996, 26(1): 29~41
- [22] Gambardella L M and Dorigo M. Solving Symmetric and Asymmetric TSPs by Ant Colonies. In *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'96)*. IEEE Press, 1996. 622~627
- [23] Dorigo M and Gambardella L M. Ant colonies for the traveling salesman problem. *BioSystems*, 1997, 43: 73~81
- [24] 郝晋, 石立宝, 周家启. 求解复杂 TSP 问题的随机扰动蚁群算法. *系统工程理论与实践*, 2002, 9: 88~91
- [25] 黄席樾, 胡小兵. 蚁群算法在 K-TSP 问题中的应用. *计算机仿真*, 2004, 21(12): 162~164
- [26] 吴斌, 史忠植. 一种基于蚁群算法的 TSP 问题分段求解算法. *计算机学报*, 2001, 24(12): 1328~1333
- [27] 马良, 蒋霞. 多目标旅行售货员问题的蚂蚁算法求解. *系统工程理论方法应用*, 1999, 8(4): 23~27
- [28] Gambardella L M, Taillard É D, and Dorigo M. Ant colonies for the QAP. *Journal of the Operational Research Society (JORS)*, 1999, 50(2): 1167~1176

- [29] Maniezzo V, Colorni A, and Dorigo M. The Ant System Applied to the Quadratic Assignment Problem. Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994
- [30] Maniezzo V and Colorni A. The Ant System applied to the quadratic assignment problem. IEEE Transactions on Data and Knowledge Engineering, 1999, 11(5):769~778
- [31] Taillard É D and Gambardella L M. Adaptive Memories for the Quadratic Assignment Problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997
- [32] Gambardella L M, Taillard É D, and Dorigo M. Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society, 1999, 50(2):167~176
- [33] 马良, 王龙德. 背包问题的蚂蚁优化算法. 计算机应用, 2001, 21(8): 4~5
- [34] 胡小兵, 黄席樾. 基于蚁群优化算法的 0-1 背包问题求解. 系统工程学报, 2005.6
- [35] 陈义保, 姚建初, 钟毅芳等. 基于蚁群系统的工件排序问题的一种新算法. 系统工程学报, 2001, 17(5): 476~480
- [36] 孙新宇, 万筱宁, 孙林岩. 蚁群算法在混流装配线调度问题中的应用. 信息与控制, 2002, 31(6): 486~290
- [37] Colorni A, Dorigo M, Maniezzo V, and Trubian M. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science(JORBEL), 1994, 34: 39~53
- [38] Li Yanjun, Wu Tiejun. A Nested Hybrid Ant Colony Algorithm for Hybrid Production Scheduling Problem. 自动化学报. 2003, 29(1): 95~101
- [39] Merkle M, Middendorf M, and Schneck H. Ant colony optimization for resource-constrained project scheduling. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), San Francisco: Morgan Kaufmann Publishers, CA, 2000. 893~900
- [40] Costa D and Hertz A. Ants can color graphs. Journal of the Operational Research Society, 1997, 48: 295~305
- [41] Bullnheimer B, Hartl R F, and Strauss C. Applying the ant system to the vehicle routing problem. IN Osman I H, S Voß S Martello, and Roucairol C, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic, 1998. 109~120
- [42] Bullnheimer B, Hartl R F, and Strauss C. Applying the Ant System to the vehicle routing problem. In S. Voß S. Martello, Osman I H, and Roucairol C, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, Dordrecht, 1999. 285~296
- [43] Bullnheimer B, Hartl R F, and Strauss C. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 1999, 89: 319~328
- [44] Gambardella L M, Taillard É D, and Agazzi G. A multiple ant colony system for vehicle routing problems with time windows. In Corne D, Dorigo M, and Glover F, editors. New Ideas in Optimization. London, McGraw Hill, 1999. 63~76
- [45] 陈峻, 沈洁, 秦玲. 蚁群算法进行连续参数优化的新途径. 系统工程理论与实践, 2003, 3: 48~53
- [46] 陈峻, 沈洁, 秦玲. 蚁群算法求解连续空间优化问题的一种新方法. 软件学报, 2002, 13(12): 2317~2323
- [47] 汪耀, 吴启迪. 蚁群算法在连续空间寻优问题求解中的应用. 控制与决策, 2003, 18(1): 45~48
- [48] 林德, 朱文兴. 凸整数规划问题的混合蚁群算法. 福州大学学报(自然科学版), 1999, 27(6): 5~9
- [49] 汪耀, 吴启迪. 蚁群算法在系统辨识中的应用. 自动化学报, 2003, 29(1): 102~109
- [50] 金飞虎, 洪炳焰, 高庆吉. 基于蚁群算法的自由飞行空间机器人路径规划. 机器人, 2002, 24(6): 526~529
- [51] 胡小兵, 黄席樾. 基于蚂蚁算法的三维空间机器人路径规划研究. 重庆大学学报(自然科学版), 2004, 27(8):132~135
- [52] 胡小兵, 黄席樾. 蚁群算法在迷宫最优路径问题中的应用. 计算机仿真, 2005, 22(2):26~29
- [53] Rafael S Parpinelli, Heitor S Lopes, and Alex A Freitas. Data Mining with an Ant Colony Optimization Algorithm. IEEE Transactions on Evolutionary Computing, 2002, 6(4): 321~332

- [54] Di Caro G and Dorigo M. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 1998, 9: 317~365
- [55] Bonabeau E, Henaux F, Guérin S, Snyers D, Kuntz P, and Theraulaz G. Routing in telecommunication networks with "Smart" ant-like agents. In *Proceedings of IATA'98, Second Int. Workshop on Intelligent Agents for Telecommunication Applications*. *Lectures Notes in AI* vol. 1437, Springer Verlag, 1998.
- [56] Di Caro G and Dorigo M. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997
- [57] G. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In Pan Y, Akl S G, and Li K, editors. *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*. Anaheim: IASTED/ACTA Press, 1998. 541~546
- [58] Heusse M, Guérin S, Snyers D, and Kuntz P. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1998, 1(2-3): 237~254
- [59] Subramanian D, Druschel P, and Chen J. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1997. 832~838
- [60] Van der Put R. Routing in the faxfactory using mobile agents. Technical Report R&D-SV-98-276, KPN Research, 1998
- [61] 吕国英, 刘泽民, 周正. 基于蚂蚁算法的分布式 QoS 路由选择算法. *通信学报*, 2001, 22(9): 35~42
- [62] 张素兵, 刘泽民. 基于蚂蚁算法的分级 QoS 路由调度方法. *北京邮电大学学报*, 2000, 23(4): 12~15
- [63] 张素兵, 刘泽民. 基于蚂蚁算法的时延受限分布式多播路由研究. *通信学报*, 2001, 22(3): 71~74
- [64] 王颖, 谢剑英. 一种基于蚁群算法的多媒体网络多播路由算法. *上海交通大学学报*, 2002, 36(4): 526~531
- [65] Lu Guoying, Liu Zemin. QoS Multicast Routing Based on Ant Algorithm in Internet. *The Journal of China Universities of Posts and Telecommunication*. 2000, 7(4): 12~17
- [66] 庄昌文, 范明钰, 李春辉, 虞厥邦. 基于协同工作方式的一种蚁群布线系统. *半导体学报*, 1999, 20(5): 401~406
- [67] Deneubourg J L, Pasteels J M, Verhaeghe J C. Probabilistic Behaviour in Ants: a Strategy of Errors. *Journal of Theoretical Biology*, 1983, 105: 259~271
- [68] Deneubourg J L, Goss S. Collective patterns and decision-making. *Ethology, Ecology & Evolution*, 1989, 1: 295~311
- [69] Goss S, Beckers R, Deneubourg J L, Aron S, Pasteels J M. How Trail Laying and Trail Following Can Solve Foraging Problems for Ant Colonies. in *Behavioural Mechanisms of Food Selection*, R. N. Hughes ed., NATO-ASI Series, vol. G 20, Berlin: Springer-Verlag, 1990
- [70] Deneubourg J L, Aron S, Goss S, and Pasteels J M. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 1990, 3: 159~168
- [71] Goss S, Aron S, Deneubourg J L, and Pasteels J M. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 1989, 76: 579~581
- [72] Pasteels J M, Deneubourg J L, and Goss S. Self-organization mechanisms in ant societies (i): Trail recruitment to newly discovered food sources. *Experientia Supplementum*, 1987, 54:155~175
- [73] Watkins C. Learning with delayed rewards. England: Psychology Department, University of Cambridge, 1989

- [74] Gambardella L M and Dorigo M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Prieditis A and Russell S, editors. *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*. Palo Alto, CA: Morgan Kaufmann Publishers, 1995. 252~260
- [75] Zlochin M, Birattari M, Meuleau N, Dorigo M. Model-based search for combinatorial optimization. Technical Report TR/IRIDIA/2001-15, IRIDIA, Université Libre de Bruxelles, 2001
- [76] Zlochin M and Dorigo M. Model-based search for combinatorial optimization: A comparative study. In *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science*. Springer Verlag, Berlin, Germany, 2002
- [77] Blum C, and Sampels M. When model bias is stronger than selection pressure. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN'02)*, Incs2439, pp.893-902. Springer Verlag, Berlin, Germany, 2002
- [78] Grassé P P. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis* et *cubitermes* sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 1959, 6: 41~81
- [79] 肖刚, 李天伦. 系统可靠性分析中的蒙特卡罗方法. 北京: 科学出版社, 2003. 300
- [80] 张乃尧, 阎平凡. 神经网络与模糊控制. 北京: 清华大学出版社, 2000. 261
- [81] 靳蕃. 神经计算智能基础——原理·方法. 成都: 西南交通大学出版社, 2000. 455
- [82] 潘正军, 康立山, 陈毓屏. 演化计算. 清华大学出版社 & 广西科学技术出版社, 1998. 203
- [83] Narendra K and Thathachar M. *Learning Automata: An Introduction*. Prentice Hall, 1989. 476
- [84] Dorigo M and Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53~66
- [85] Stützle T and Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem. In Baack T, Michalewicz Z, and Yao X, editors. *Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference*, IEEE Press, 1997. 309~314
- [86] Stützle T and Hoos H. Improvements on the ant system: Introducing MAX-MIN ant system. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Springer Verlag, Wien, 1997. 245~249
- [87] Stützle T and Hoos H. MAX-MIN Ant system and local search for combinatorial optimization problems. In Voß S, Martello S, Osman I H, and Roucairol C, editors. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Boston: Kluwer, 1998. 137~154
- [88] Stützle T. MAX-MIN Ant System for Quadratic Assignment Problems. Technical Report AIDA-97-04, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germany, July 1997
- [89] Bullnheimer B, Hartl R F, and Strauss C. A new rank-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, 1999, 7(1):25~38
- [90] 吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法. *计算机研究与发展*. 1999, 36(10): 1240~1245
- [91] 王颖, 谢剑英. 一种自适应蚁群算法及其仿真研究. *系统仿真学报*. 2002, 14(1): 32~33
- [92] 覃刚力, 杨家本. 自适应调整信息素的蚁群算法. *信息与控制*. 2002, 31(3): 198~201

- [93] Schoonderwoerd R, Holland O, Bruten J, and Rothkrantz L. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 1996, 5(2): 169~207
- [94] Schoonderwoerd R, Holland O, and Bruten J. Ant-like agents for load balancing in telecommunications networks. In *Proceedings of the First International Conference on Autonomous Agents*. ACM Press, 1997. 209~216
- [95] White T, Paturek B, and Oppacher F. Connection management using adaptive mobile agents. In *Arabnia H R, editor. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*. CSREA Press, 1998. 802~809
- [96] Stützle T. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*. Infix, Sankt Augustin, Germany, 1999
- [97] Stützle T and Hoos H H. MAX-MIN Ant System. *Future Generation Computer Systems*, 2000, 16(8): 889~914.
- [98] Cordon O, Fernández de Viana I, Herrera F, and Moreno L. A new ACO model integrating evolutionary computation concepts: The best-worst ant system. In *Dorigo M, Middendorf M, and Stützle T, editors. Abstract proceedings of ANTS2000- From Ant Colonies to Artificial Ants: A Series of International Workshops on Ant Algorithms*. Universit Librede Bruxelles, 2000. 22~29
- [99] Maniezzo V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 1999, 11(4): 358~369
- [100] Stützle T. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3. Mainz, Aachen: Verlag, 1998. 560~1564
- [101] Bauer A, Bullnheimer B, Hartl R F, and Strauss C. An ant colony optimization approach for the single machine total tardiness problem. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*. Piscataway, NJ: IEEE Press, 1999. 1445~1450
- [102] Den Besten M, Stützle T, and Dorigo M. Ant colony optimization for the total weighted tardiness problem. In *Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo J J, and Schwefel H S, editors. Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer Verlag, 2000. 611~620
- [103] Schoonderwoerd R, Holland O, Bruten J, and Rothkrantz L. Antbased load balancing in telecommunications networks. *Adaptive Behavior*, 1996, 5(2): 169~207
- [104] Gambardella L M and Dorigo M. HAS-SOP: An hybrid Ant System for the sequential ordering problem. *Technical Report IDSIA-11-97*, IDSIA, Lugano, Switzerland, 1997
- [105] Gambardella L M and Dorigo M. Ant Colony System hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 2000, 12(3): 237~255
- [106] Michel R and Middendorf M. An island model based Ant System with lookahead for the shortest supersequence problem. In *Eiben A E, Bäck T, Schoenauer M, and Schwefel H P, editors. Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer Verlag, 1998. 692~701
- [107] Michel R and Middendorf M. An ACO algorithm for the shortest supersequence problem. In *Corne D, Dorigo M, and Glover F, editors. New Ideas in Optimization*, London, UK: McGraw Hill, 1999. 51~61

- [108] Maniezzo V and Carbonaro A. An ANTS heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 2000, 16(8): 927~935
- [109] Leguizamón G and Michalewicz Z. A new version of Ant System for subset problems. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, Piscataway, NJ: IEEE Press, 1999. 1459~1464
- [110] Navarro Varela G and Sinclair M C. Ant colony optimization for virtual wave length path routing and wavelength allocation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*. Piscataway, NJ: IEEE Press, 1999. 1809~1816
- [111] Liang Y C and Smith A E. An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP). *IEEE Transactions on Reliability*, 2004, 53(3): 417~423
- [112] Solnon C. Solving permutation constraint satisfaction problems with artificial ants. In Horn W, editor. *Proceedings of the 14th European Conference on Artificial Intelligence*. Amsterdam: IOS Press, 2000. 118~122

第十章 蚁群优化元启发式及其收敛性

第一节 引言

经过几年的发展, ACO 在许多领域得到了广泛的应用^[1]. 大多数 ACO 算法都是在 AS 算法及其改进算法的基础上结合具体问题而开发的, 因此不具有通用性. 这样, 设计出一种如遗传算法^[2,3]中编码、选择、交叉、变异的通用框架变得极为迫切. 1999 年, M Dorigo^[1,4]等人提出了蚁群优化元启发式 (ant colony optimization meta heuristic, 简称 ACO-MH), 为 ACO 算法的理论研究和算法设计提供了一个统一的框架.

另一方面, 对 ACO 的收敛性研究也取得了一些初步的结论. 1999 年, W J Gutjahr^[5]首次提出了基于图的蚂蚁系统 (graph-based ant system, 简称 GBAS) 元启发式这一 ACO 的通用模型, 并证明在一定的条件下 GBAS 能以任意接近 1 的概率收敛到最优解. 之后, W J Gutjahr^[6,7]又将 GBAS 发展成 GBAS/tde v 和 GBAS/tdlb, 并证明了这两个算法能以概率 1 收敛到最优解. T Stützle 和 M Dorigo^[8]对一类被称为 $ACO_{\tau_{min}}$ 的 ACO 算法的收敛性进行了研究, 并证明这类 ACO 算法在迭代次数足够大时能以任意接近 1 的概率收敛到最优解, 该结论可以直接应用到两类实验上最成功的 ACO 算法——MMAS 和 ACS. N Meuleau 和 M Dorigo^[9]研究了随机梯度下降 (stochastic gradient descent, 简称 SGD) 和 ACO 之间的关系, 将 ACO 看成是一种近似的 SGD 算法, 并根据 SGD 实现了理论上收敛的 ACO 算法.

ACO-MH 和 ACO 的收敛性结论为 ACO 算法的研究奠定了初步的理论基础, 本章就这两方面的内容进行了论述.

第二节 蚁群优化元启发式

一、人工蚂蚁与真实蚂蚁的异同

ACO 是一种基于群体的、用于求解复杂优化问题的通用搜索技术. 与真实蚂蚁通过外激素的留存 / 跟随行为进行间接通讯相类似, ACO 中一群简单的人工蚂蚁 (主体) 通过信息素 (一种分布式的数字信息, 与真实蚂蚁释放的外激素相对应) 进行间接通讯, 并利用该信息和与问题相关的启发式信息逐步构造问题的解.

人工蚂蚁具有双重特性：一方面，他们是真实蚂蚁的抽象，具有真实蚂蚁的特性；另一方面，他们还有一些在真实蚂蚁中找不到的特性，这些新的特性，使人工蚂蚁在解决实际优化问题时，具有更好地搜索较好解的能力。

(一) 人工蚂蚁与真实蚂蚁的相同点

人工蚂蚁与真实蚂蚁的相同点为：

(1) 都是一群相互协作的个体。与真实蚁群一样，ACO 由一群人工蚂蚁组成，人工蚂蚁之间通过同步 / 异步协作来寻找问题的最优解。虽然单只人工蚂蚁可以构造出问题的解，但只有当多只人工蚂蚁通过相互协作，才能发现问题的最优（次优）解。人工蚂蚁个体间通过写 / 读问题的状态变量来进行协作。

(2) 都使用外激素的迹和 Stigmergy 机制。如真实蚂蚁一样，人工蚂蚁通过改变所访问过的问题的数字状态信息（在 ACO 中被称为信息素）来进行间接的协作。在 ACO 中，信息素是人工蚂蚁之间进行交流的唯一途径。这种通讯方式在群体知识的利用上起到了至关重要的作用。另外，ACO 还用到了蒸发机制，这一点对应于真实蚂蚁中外激素的蒸发现象。蒸发机制使蚁群逐渐忘记过去的历史，使后来的蚂蚁在搜索中较少地受过去较差解的影响，从而更好地指导蚂蚁的搜索方向。

(3) 搜索最短路径与局部移动。人工蚂蚁和真实蚂蚁具有相同的任务，即以局部移动的方式构造出从原点（蚁巢）到目的点（食物源）之间的最短路径。

(4) 随机状态转移策略。人工蚂蚁和真实蚂蚁都按照概率决策规则从一种状态转移到另一种邻接状态。其中的概率决策规则是与问题相关信息和局部环境信息的函数。在状态转移过程中，人工蚂蚁与真实蚂蚁都只用到了局部信息，没有使用前瞻策略来预见将来的状态。因此，所使用的策略在时间和空间上是局部的。

(二) 人工蚂蚁与真实蚂蚁的不同点

人工蚂蚁与真实蚂蚁的不同点为：

(1) 人工蚂蚁生活在离散的世界，从一种离散状态到另一种离散状态；

(2) 人工蚂蚁具有内部状态，即人工蚂蚁具有一定的记忆能力，能记住自己所走过的地方；

(3) 人工蚂蚁释放信息素的数量是其生成解的质量的函数；

(4) 人工蚂蚁更新信息素的时机依赖于特定的问题。例如，大多数人工蚂蚁仅仅在蚂蚁找到一个解之后才更新路径上的信息素。

有时，为了提高 ACO 的效率，可以在算法中加入预测、局部优化以及后退等策略。

二、蚁群优化元启发式

(一) 问题的表示

在用 ACO-MH 求解离散优化问题时, 首先要将待求解问题表示成具有如下特征的构造图的形式:

- (1) 给定一组有限元素的集合 $C = \{c_1, c_2, \dots, c_{N_c}\}$;
- (2) 设 $\bar{C} \subseteq C \times C$, 并定义可能的连接 (转移) 有限集 $L = \{l_{c_i c_j} \mid (c_i, c_j) \in \bar{C}\}$, $|L| \leq N_c^2$;
- (3) 对每一个连接 $l_{c_i c_j} \in L$, 定义连接费用函数 $J_{c_i c_j} = J(l_{c_i c_j}, t)$, 其中 t 为时间参数;

(4) 定义集合 C 和集合 L 中元素的约束条件构成的有限集 $\Omega = \Omega(C, L, t)$;

(5) 问题的状态定义为集合 C 中的序列 $s = \langle c_i, c_i, \dots, c_k, \dots \rangle$. 设 S 为所有可能的序列组成的集合, \tilde{S} 为所有满足约束条件 $\Omega(C, L, t)$ 的序列集合, 且 $\tilde{S} \subseteq S$, 则集合 \tilde{S} 定义了问题的所有可行状态. 序列 s 的长度 (即序列 s 中元素的个数) 表示为 $|s|$;

(6) 邻域结构定义如下: 状态 s_2 被认为是状态 s_1 的邻域, 如果 (i) $s_1, s_2 \in \tilde{S}$; (ii) 经过一个逻辑步长, 可以从状态 s_1 到达状态 s_2 . 即如果 c_1 为序列 s_1 的最后一个元素 (Component), 则一定存在 $c_2 \in C$ 使得 $l_{c_1 c_2} \in L$, $s_2 \equiv \langle s_1, c_2 \rangle$;

(7) 解 ψ 是满足问题所有条件的集合 \tilde{S} 中的一个元素. 多维解是指集合 C 上元素的多个独立的序列集;

(8) 每一个解 ψ 与一个费用函数 $J_\psi(L, t)$ 相关, $J_\psi(L, t)$ 是解 ψ 中所有连接的费用 $J_{c_i c_j}$ 的函数.

对于一个给定的离散优化问题, 可以将其表示为构造图 $G = (N, L)$ 的形式, 优化问题的解可表达为 G 中的可行路径. ACO-MH 主要用来求解带约束条件的最小费用路径问题.

在 ACO-MH 中, 一群人工蚂蚁在与优化问题相关的构造图中通过间接通讯与协作的方式搜索问题的最优解. 蚂蚁在搜索解的过程中将收集到的信息编码成与连接 l_{ij} 相关的信息素 τ_{ij} . 信息素是蚂蚁在整个搜索过程的长期积累的内存信息, 与问题的表示有关. 信息素可与图 G 中的所有弧或只与其中的部分弧相关. 图 G 中的弧 ij 还可能与表示问题先验知识的启发式信息 η_{ij} 相关.

(二) 蚁群的一般特征

在 ACO-MH 中, 蚁群具有如下的一般特征:

(1) 虽然单只蚂蚁能够找到问题的一个可行解, 但高质量的解是在多只蚂蚁 (蚁群) 的集体交互 / 协作中产生的;

(2) 每只蚂蚁只使用自己的私有信息和其所处节点的局部信息;

- (3) 蚂蚁间通过读 / 写存储在信息素变量中的信息进行间接通讯;
- (4) 蚂蚁本身不具有自适应性, 相反地, 他们自适应地修改问题的表示方式.

(三) 人工蚂蚁的行为

在 ACO-MH 中, 人工蚂蚁具有如下的行为特征:

- (1) 每只蚂蚁搜索最小费用的可行解 $\bar{J}_\psi = \min_\psi J_\psi(L, t)$;
- (2) 蚂蚁 k 的内存表 M^k 记录其走过的路径的信息, M^k 的作用为: (i) 生成可行解; (ii) 用来评估发现的解; (iii) 蚂蚁在沿原路往回走时使用, 以便更新路径上的信息素;
- (3) 处于状态 $s_r = \langle s_{r-1}, i \rangle$ 的蚂蚁 k 根据其可行邻域 N_i^k 能够移动到节点 j , 其中 $N_i^k = \{j | (j \in N_i) \wedge (\langle s_r, j \rangle \in \bar{S})\}$;
- (4) 蚂蚁 k 有一个起始状态 s_s^k 和一个或多个终止条件 e^k ;
- (5) 蚂蚁从起始状态出发, 在可行邻域状态中逐步移动, 以增量方式构造问题的可行解, 当至少有一个终止条件满足时, 蚂蚁构造解的过程结束;
- (6) 对应于节点 i 的蚂蚁 k , 通过随机决策规则移动到邻域 N_i^k 中的一个节点 j ;
- (7) 蚂蚁的随机决策规则与下列因素相关: (i) 存储在节点中的被称为蚂蚁路由表的局部数据结构值, 该值是局部信息素和启发式信息的函数; (ii) 存储蚂蚁过去历史信息的私有内存信息; (iii) 问题的约束条件;
- (8) 当从节点 i 移动到邻域节点 j 后, 蚂蚁可以更新弧 (i, j) 上的信息素 τ_{ij} , 这种更新方式称为在线逐步信息素更新 (online step-by-step pheromone update);
- (9) 蚂蚁生成问题的解后, 沿原路返回并在往回走的过程中更新其走过的路径上的信息素, 该过程称为在线延迟信息素更新 (online delayed pheromone update);
- (10) 一旦生成问题的解之后, 蚂蚁沿原路走回初始状态节点, 并释放所有已分配的资源, 然后死去.

在 ACO-MH 中, 一群人工蚂蚁通过局部随机决策规则和节点的局部路由表在问题的状态空间中按同步 / 异步方式移动, 并以增量方式构造问题的可行解, 一旦构造可行解的过程完成, 蚂蚁将对其进行评估, 同时根据解的质量的好坏释放相应数量的信息素到其走过的边上, 该信息素将指导后来蚂蚁的搜索行为.

除此之外, ACO-MH 中还增加了两个过程: 信息素蒸发机制与集中控制过程 (Daemon action). 信息素蒸发机制即弧上信息素的浓度随时间自动减少的过程, 该过程可避免算法快速收敛于问题的局部最优解. 通过忘记过去的部分行为, 蚂蚁可以更好地探索问题中新的解空间. 集中控制过程用来处理单只蚂蚁不能处理的行为, 如局部优化行为, 或是从全局的观点来看, 全局信息对于指导蚂蚁搜索解的过程是否有效. 在实际应用中, 集中控制过程可以通过观察每只蚂蚁找到的路径并选择在最优路径上释放额外的信息素. 算法 ACO-MH 的伪代码如下:

algorithm ACO-MH

```

while (未满足停止条件)
    schedule_activities
        ants_generation_and_activity(); { 产生蚂蚁并使其开始活动 }
        pheromone_evaporation(); { 信息素蒸发 }
        daemon_actions(); { 集中控制过程, 可选 }
    end
end
end

```

算法 ACO-MH 主要由三个过程组成. 在每次迭代开始, 首先生成蚂蚁并使其处于活动状态 (通过 `ants_generation_and_activity` 过程), 当蚂蚁完成生命周期后, 对信息素进行蒸发操作 (通过 `pheromone_evaporation` 过程), 最后是可选的集中控制行为 (通过 `daemon_actions` 过程). 下面是过程 `ants_generation_and_activity` 的伪代码:

```

procedure ants_generation_and_activity()
    while (有资源可利用)
        schedule_the_creation_of_a_new_ant(); { 创建新的蚂蚁 }
        new_active_ant(); { 蚂蚁生命周期的活动 }
    end
end

```

蚂蚁生命周期活动 `new_active_ant` 的伪代码如下:

```

procedure new_active_ant()
    initialize_ant(); { 初始化蚂蚁 }
    M = update_ant_memory(); { 更新蚂蚁的内存信息, 并存于 M 中 }
    while(当前状态 ≠ 目标状态)
        A = read_local_ant_routing_table();
        { 读取蚂蚁局部路由表信息, 并存于 A 中 }
        P = compute_transition_probabilities(A; M; 问题约束条件);
        { 通过 A、M 问题约束条件计算转移概率 P }
        Next_state = apply_ant_decision_policy(P; 问题约束条件);
        { 通过 P 和问题约束条件选择下一个状态 }
        Move_to_next_state(next_state); { 蚂蚁移动到下一个状态 }
        if(使用在线逐步信息素更新策略)
            deposit_pheromone_on_the_visited_arc();
            { 更新已访问弧上的信息素 }
        update_ant_routing_table();
        { 更新蚂蚁的路由表 }
    end
end

```

```

end if
M= update_internal_state(); { 更新内部状态 }
end
if (使用在线延迟信息素更新策略)
    evaluate_solution(); { 评估解 }
    deposit_pheromone_on_all_visited_arcs();
    { 更新所有已访问弧上的信息素 }
    update_ant_routing_table(); { 更新蚂蚁的路由表 }
end
die(); { 结束蚂蚁的生命周期 }
end

```

在 ACO-MH 中, 蚂蚁何时释放信息素以及释放多少取决于特定问题的具体实现. 蚂蚁可以在生成解的过程中逐步的释放信息素, 也可以等产生一个完整的解后再沿原路往回走的过程中释放, 或者两者同时使用.

自催化机制在 ACO-MH 中扮演了重要的角色. 如果较多蚂蚁选择了某一条弧, 则将有更多的信息素放置到其上, 从而使得更多的蚂蚁对其感兴趣. 因此, 如果蚂蚁的某一次状态转移有利于产生一个高质量的解, 其对环境的影响也会更大.

环境信息与启发式相结合构成了蚂蚁路由决策表 (一个概率表, 指导蚂蚁的搜索行为). 随机移动策略和信息素蒸发机制使蚂蚁很快地进入最优 (次优) 解空间的某一部分.

蚂蚁完成任务后 (即生成问题的一个可行解, 更新了信息素) 将死去, 并从系统中删除与该蚂蚁有关的资源.

在 ACO-MH 中还可以包含全局控制操作 `dae.actions`, 该操作可以在更新信息时利用附加的全局信息, 为可选的.

ACO-MH 中可能还需要一些同步, 由 `schedule.activities` 来处理. 该调度对分布式问题尤其重要.

三、解的评估、信息素及蚂蚁数对 ACO-MH 的影响

(一) 解的隐式 / 显式评估

在 ACO-MH 中, 蚂蚁生成的解以反馈方式指导后来蚂蚁的搜索行为. 该过程通过两个机制来实现. 第一种机制是解的显式评估, 即通过对解的好坏的评价来决定蚂蚁释放信息素的多少, 所有 ACO-MH 都使用该机制; 第二种机制是解的隐式评估, 该机制利用不同长度路径对蚂蚁觅食行为的影响, 即经过较短路径的蚂蚁将先释放信息素, 从而对后来的蚂蚁产生较大的影响. 实验证明, 对于像通讯网络等

地理上分布的问题,解的隐式评估具有非常重要的作用.文献[10, 11]在只使用解的隐式评估的情况下,仍然能够找到问题的最优解.当然,如果将解的隐式/显式评估组合起来后,可以大大提高 ACO-MH 的性能.

(二) 解的显式评估和信息素留存

蚂蚁在生成一个可行解后,便使用其评估应该释放多少信息素.例如,在 AS 算法中,蚂蚁在其经过的边上释放的信息素量与解的值成反比.这仅仅是众多显式评估方式中的一种.对于像网络路由问题,或其他动态问题(在求解过程中,问题的特性在不断地发生变化,且不可预见),则没有简单的方法来评估解和决定在所经过的边上释放多少信息素.在 AntNet^[10,11]中,蚂蚁在线学习网络状态模型,并利用该信息评估解的质量的好坏.

(三) 蚂蚁数

ACO-MH 中的蚂蚁数必须通过实验来确定.庆幸的是,在蚂蚁数变化较大的情况下,ACO 仍然非常健壮.既然如此,在 ACO 中为什么要使用一群蚂蚁(即蚂蚁数 $m > 1$),而不使用单只蚂蚁呢?原因是,虽然单只蚂蚁也可以产生问题的解,但出于对效率 (efficiency) 的考虑,还是倾向于使用一群蚂蚁.对于分布式的图形问题更应如此.因为对不同长度解的利用只能在一群蚂蚁中才能体现出来.另一方面,在组合优化问题中,蚂蚁同步的移动, m 只蚂蚁生成 NC 个解(其中 NC 为蚂蚁的迭代次数)等同于单只蚂蚁生成 $m \cdot NC$ 个解.但实验结果显示,当蚂蚁数设置为 $m > 1$ 时,算法有更好的性能.

第三节 蚁群优化的收敛性

一、蚁群优化收敛性概述

W J Gutjahr^[5](1999)首先对 ACO 的收敛性进行了研究,并提出了基于图的蚂蚁系统 (GBAS). GBAS 是一种更为特殊的 ACO-MH 模型,适用于各种静态组合优化问题.在 GBAS 中,设 P_t 表示某只蚂蚁在第 t 次迭代时找到最优路径的概率,则关于 GBAS 的收敛性有如下结论:(i) 对任意的 $\varepsilon (0 < \varepsilon < 1)$ 和固定的蒸发系数 $\rho (0 < \rho < 1)$,如果蚂蚁数 S 足够大,则对任意的 $t \geq t_0 (t = t_0(\varepsilon))$ 有 $P_t \geq 1 - \varepsilon$ 成立;(ii) 对任意的 $\varepsilon (0 < \varepsilon < 1)$ 和固定数量的蚂蚁,如果蒸发系数 $\rho (0 < \rho < 1)$ 足够接近 0,则对任意的 $t \geq t_0 (t = t_0(\varepsilon))$ 有 $P_t \geq 1 - \varepsilon$ 成立.以上结论中的 ε 可以任意接近 0,但对于某个给定的 ε ,相应的蚂蚁数或者蒸发系数不能确定,因此,对于某个事先给定的概率,无法确定蚂蚁数 S 和蒸发系数 ρ .从这个意义上讲,GBAS 是不可控的.在文献[6, 7]中, Gutjahr 将 GBAS 发展成 GBAS/tdev 和 GBAS/tldb,

并证明其能以概率 1 收敛到问题的最优解。T Stützle 和 M Dorigo^[8](2001) 对一类称为 $\text{ACO}_{\tau_{\min}}$ 的 ACO 的收敛性进行了研究, 证明了对任意 $\varepsilon > 0$, 算法发现最优解至少一次的概率 $P^*(t) \geq 1 - \varepsilon$, 并且当 $t \rightarrow \infty$ 时, $P^*(t)$ 趋于 1. 该收敛性结论可直接应用于 MMAS 算法和 ACS 算法. Meleau 和 M Dorigo^[9] 研究了随机梯度下降形式的 AS 算法.

二、保证收敛到最优解的 ACO 算法^[12]

(一) 算法描述

定义 10.3.1 给定一个组合优化问题的实例, 则与该实例相关的有向图 $C = (V, A)$ 和映射 ϕ 具有如下的性质:

- (1) 在图 C 中, 有一个唯一的起始节点;
- (2) 设 W 是图 C 中 (有向) 路径 w 的集合, 且满足如下条件:

- i) w 以图 C 的起始节点为起点;
- ii) w 最多包含图 C 中的节点一次;
- iii) w 的最后节点不存在后继. 即在不违背条件 ii) 的情况下, 图 C 中除掉路径 w 上的节点后不存在路径 w 的后继节点.

其中 ϕ 将集合 W 的一个子集 \overline{W} 映射到给定问题的可行解空间, 即对于 \overline{W} 中的每一条路径 w , 通过 ϕ 可得到一个问题的可行解; 反过来, 问题的一个可行解通过逆映射 ϕ^{-1} 可得到集合 \overline{W} 中的至少一条路径.

基于定义 10.3.1, GBAS/tded 和 GBAS/tdlb 可描述如下:

初始化构造图 C 中弧 (k, l) 上信息素的迹 τ_{kl} ;

for 迭代次数 $n=1, 2, \dots$ {

 for 蚂蚁 $s=1, 2, \dots, S$ {

 将蚂蚁 s 的当前位置 k 设置为图 C 的起始节点;

 将蚂蚁 s 的当前路径 u 设置为空表;

 while (蚂蚁 s 的路径 u 的可行后继延长 (k, l) 存在) {

- 1) 蚂蚁以概率 p_{kl} 选择后继节点 l , 如果延长 (k, l) 不可行, 则 $p_{kl} = 0$; 否则 $p_{kl} = \tau_{kl} / \left(\sum_{(k,r)} \tau_{k,r} \right)$, 其中 $\left(\sum_{(k,r)} \tau_{k,r} \right)$ 是对所有可行路径上的信息素求和;
- 2) 将蚂蚁 s 的路径 u 延长到 l , 设置 $k := l$.

 } //end while

} //end for

更新信息素;

} //end for

此处, 部分路径 u 的延长 (k, l) 的可行性由定义 10.3.1 中的性质 (2) 确定. 蚂蚁在选择路径时没有考虑启发式因子, 这样可以增加表示的透明性, 并且使该证明可以扩展到非常数启发式因子的情况. 在该算法中, 根据信息素更新规则的不同, 可以产生两类 GBAS 算法, 即时间相关蒸发系数的 GBAS(GBAS with time-dependent evaporation factor, 简称 GBAS/tdev) 和时间相关信息素下界的 GBAS(GBAS with time-dependent lower pheromone bound, 简称 GBAS/tdlb). 初始时刻, GBAS/tdev 和 GBAS/tdlb 将所有边上信息素都初始化为 $\tau_{kl} = \tau_{kl}(1) = 1/|A|$, 其中 $|A|$ 为图 C 中弧的数目.

GBAS/tdev 中信息素更新规则为

$$\tau_{kl}(n+1) := \begin{cases} (1 - \rho_n) \tau_{kl}(n) + \rho_n / L(\hat{w}(n)), & \text{如果 } (k, l) \in \hat{w}(n), \\ (1 - \rho_n) \tau_{kl}(n), & \text{否则,} \end{cases} \quad (10.3.1)$$

其中, $0 < \rho_n < 1 (n = 1, 2, \dots)$. $\hat{w}(n)$ 是第 n 次迭代后发现的最好路径, 该路经存储在—个列表中, 每当有蚂蚁发现较好路径时, 就用较好路经替换该列表中的值. $L(w)$ 是路径 w 的长度, 即路径 w 中弧的数目. GBAS/tdev 的主要特点是蒸发系数 ρ_n 为时间相关的, 即 ρ_n 的值是迭代次数 n 的函数.

GBAS/tdlb 中信息素更新规则为

$$\tau_{kl}(n+1) := \begin{cases} \max \left((1 - \rho) \tau_{kl}(n) + \rho / L(\hat{w}(n)), \tau_{\min}(n) \right), & \text{如果 } (k, l) \in \hat{w}(n), \\ \max \left((1 - \rho) \tau_{kl}(n), \tau_{\min}(n) \right), & \text{否则,} \end{cases} \quad (10.3.2)$$

其中, $\rho (0 < \rho < 1)$ 为常数. $\tau_{\min}(n)$ 是一个实数序列, 其他参数同上. 信息素的迹 $\tau_{\min}(n)$ 有界且与时间相关. 与 GBAS/tdev 中所有信息素之和为 1 不同, GBAS/tdlb 中所有信息素之和为变化的.

(二) 随机过程描述

考虑算法执行过程中的如下两个状态变量:

- 1) 向量 $\underline{\tau}(n)$ 表示第 n 次迭代开始时, 图 C 中所有弧上信息素的值;
- 2) 向量 $\hat{w}(n-1)$ 表示第 n 次迭代开始时 (第 $n-1$ 次迭代结束时), 算法找到的最好路径, 其中 $\hat{w}(0)$ 可以定义为任意可行路径.

这里 $\underline{\tau}(n)$ 是一个由 $|A|$ 个元素组成的向量, $\hat{w}(n-1)$ 可以表示成最多由 $|V|$ 个元素组成的向量, 其中 $|V|$ 为图 C 的节点总数. 由于 \overline{W} 中共有有限多个元素, 故可用一个整数来标识 $\hat{w}(n-1)$. 因此 $(\underline{\tau}(n), \hat{w}(n-1))$ 是 $IR^{|A|} \times IN$ 中的一个元素. 下面的引理将以上的随机过程描述成马尔科夫过程.

引理 10.3.1 在 GBAS/tdev 和 GBAS/tdlb 中, 由状态

$$X_n = (\underline{\tau}(n), \hat{w}(n-1)) \quad (1, 2, \dots)$$

所构成的随机过程是离散时间空间上的非齐次马尔科夫过程。

证明 由马尔科夫过程的性质可知, 只要证明状态 X_n 的分布仅仅依赖于状态 X_{n-1} . 不难发现, 在第 $n-1$ 次迭代中, 蚂蚁 $s(s=1, 2, \dots, S)$ 生成的路径 $w^{(s)}(n-1)$ 的分布仅仅依赖于 $\tau(n-1)$, 且这些路径和 $\hat{w}(n-2)$ 一起决定 $\hat{w}(n-1)$, 而 $\hat{w}(n-1)$ 又通过信息素的更新来决定 $\tau(n)$. 因此, $(\tau(n), \hat{w}(n-1))$ 的分布由 $(\tau(n-1), \hat{w}(n-2))$ 完全决定. 由于信息素更新依赖于 n , 故该马尔科夫过程为非齐次的. ■

值得注意的是, $(\tau(n))$ 不是马尔科夫过程, 因为在没有第二项的情况下, 他不是自包容的 (self-contained).

(三) 相关结果

定理 10.3.1 在算法 GBAS/tdev 中, 设

$$\rho_n \leq 1 - \frac{\log n}{\log(n+1)} \quad (n \geq N) \quad (10.3.3)$$

对某些 $N \geq 1$ 成立, 并且

$$\sum_{n=1}^{\infty} \rho_n = \infty, \quad (10.3.4)$$

则当 $n \rightarrow \infty$ 时, 马尔科夫过程 $X_n = (\tau(n), \hat{w}(n-1))$ ($1, 2, \dots$) 以概率 1 收敛到状态 $(\tau[w^*], w^*)$, 其中 w^* 是一条最优路径, 且 $\tau[w^*]$ 定义为

$$\tau_{kl}[w^*] = \begin{cases} 1/L(w^*), & \text{如果 } (k, l) \in w^*, \\ 0, & \text{否则.} \end{cases} \quad (10.3.5)$$

特别地, 在满足式 (10.3.3), (10.3.4) 的情况下, 设某一固定蚂蚁 s 在第 n 次迭代时经过最优路径 w^* 的概率 $p_{w^*}(n)$, 则当 $n \rightarrow \infty$ 时, $p_{w^*}(n)$ 趋向于 1.

证明 (1) 定义事件 F_n 为第 n 次迭代时, 某只蚂蚁首次经过最优路径. 考虑某一条固定的最优路径 w^* , 有

$$\neg F_1 \wedge \neg F_2 \wedge \dots \implies \text{没有任何蚂蚁经过最优路径 } w^*,$$

因此有

$$P(\neg F_1 \wedge \neg F_2 \wedge \dots) \leq P(\text{没有任何蚂蚁经过最优路径 } w^*) = \prod_{n=1}^{\infty} P(\text{第 } n \text{ 次迭代没找到最优路径 } w^* | \text{第 } i(i < n) \text{ 次迭代没找到最优路径 } w^*). \quad (10.3.6)$$

考虑第 n 次迭代结束后, 每条固定弧 (k, l) 上信息素的下界. 在最坏的情况

下, 弧 (k, l) 上的信息素只蒸发而不增强. 设 $N \geq 2$, 则对 $n \geq N$, 由式 (2.3) 得

$$\begin{aligned}\tau_{kl}(n) &= \left[\prod_{i=1}^{n-1} (1 - \rho_i) \right] \tau_{kl}(1) \geq \left[\prod_{i=1}^{N-1} (1 - \rho_i) \right] \left[\prod_{i=N}^{n-1} \frac{\log i}{\log(i+1)} \right] \tau_{kl}(1) \\ &= \left[\prod_{i=1}^{N-1} (1 - \rho_i) \right] \tau_{kl}(1) \cdot \frac{\log N}{\log n} = \frac{\text{const}}{\log n}.\end{aligned}\quad (10.3.7)$$

由于对某一固定节点, 其可行的后继节点数不会超过 $|V|$, 又因为所有信息素的迹小于等于 1, 因此

$$p_{kl}(n) \geq \frac{\text{const}}{|V| \cdot \log n} \quad (n \geq N).$$

进一步, 某一固定蚂蚁 s 在第 n ($n \geq N$) 次迭代时经过最优路径 w^* 的概率为

$$\prod_{(k,l) \in w^*} p_{kl}(n) \geq \left(\frac{\text{const}}{|V| \cdot \log n} \right)^{L(w^*)},$$

很显然, 不等式右边项的下界与第 n 次迭代前独立, 因此其估计值对迭代 $1, 2, \dots, n-1$ 时发生的任何事件有条件成立. 因此, 式 (10.3.6) 右边项的上界为

$$1 \cdot \prod_{n=N}^{\infty} \left[1 - \left(\frac{\text{const}}{|V| \cdot \log n} \right)^{L(w^*)} \right]. \quad (10.3.8)$$

对式 (10.3.8) 取对数后得

$$\sum_{n=N}^{\infty} \log \left(1 - \left(\frac{\text{const}}{|V| \cdot \log n} \right)^{L(w^*)} \right) \leq - \sum_{n=N}^{\infty} \left(\frac{\text{const}}{|V| \cdot \log n} \right)^{L(w^*)} = -\infty.$$

对每个正整数 L , 级数 $\sum_n (\log n)^{-L}$ 发散, 因此式 (10.3.8) 和式 (10.3.6) 的右边项为零. 因此证明事件 $F_1 \vee F_2 \vee \dots$ 在某次迭代时发生的概率为 1.

(2) 下面证明, 对于最优路径集中的每条路径 w^* , 状态 $(\tau[w^*], w^*)$ 是马尔科夫过程 X_n 的吸引域 $S_{|A|} \times \{w^*\}$ 上的一个确定性吸引子, 其中 $\tau[w^*]$ 由式 (10.3.5) 给定. 即如果马尔科夫过程的实现 $X_1(\omega), X_2(\omega), \dots$ 对某个 n 满足 $X_n(\omega) \in S_{|A|} \times \{w^*\}$, 则 $\lim_{n \rightarrow \infty} X_n(\omega) = (\tau[w^*], w^*)$.

设 m 表示首次发现最优解 w^* 的迭代次数. 这意味着过程 $(\tau(n), \hat{w}(n-1))$ 在第 $m+1$ 次迭代时进入集合 $S_{|A|} \times \{w^*\}$. 因此, 当 $n > m$ 时, $\hat{w}(n) = w^*$, 只有 w^* 上的信息素得到加强. 对于弧 $(k, l) \in w^*$ 利用更新公式 (10.3.1) 的上一行, 得

$$\tau_{kl}(m+l) = \left[\prod_{n=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) + \frac{1}{L(w^*)} \sum_{i=0}^{r-1} \rho_{m+i} \prod_{j=i+1}^{r-1} (1 - \rho_{m+j}). \quad (10.3.9)$$

由式 (10.3.4) 知级数 $\sum \rho_n$ 发散, 即 $\prod_{n=1}^{\infty} (1 - \rho_n) = 0$, 因此在式 (10.3.9) 中有

$$\lim_{n \rightarrow \infty} \left[\prod_{m=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) = \tau_{kl}(m) \prod_{n=m}^{\infty} (1 - \rho_n) = 0. \quad (10.3.10)$$

现在考虑弧 $(k, l) \notin w^*$ 的情况. 在第 m 次迭代后, 这些弧上的信息素没有得到任何加强, 则

$$\tau_{kl}(m+l) = \left[\prod_{n=m}^{m+r-1} (1 - \rho_n) \right] \tau_{kl}(m) \rightarrow 0 \quad (r \rightarrow \infty). \quad (10.3.11)$$

因此, 属于非最优路径的弧上的信息素之和为趋近 0. 由 GBAS/tdev 中信息素的正规化属性知, 最优路径上信息素的迹之和应该为 1. 由式 (10.3.9)、(10.3.10), 对每一个弧 $(k, l) \in w^*$, $\limsup_{r \rightarrow \infty} \tau_{kl}(m+r)$ 具有同样的值 (式 (10.3.9) 的第 2 项与 (k, l) 无关), 并且这些值之和为 1. $\liminf_{r \rightarrow \infty} \tau_{kl}(m+r)$ 也满足同样的性质. 因此, 对弧 $(k, l) \in w^*$ 有

$$\limsup_{r \rightarrow \infty} \tau_{kl}(m+r) = \liminf_{r \rightarrow \infty} \tau_{kl}(m+r) = \lim_{r \rightarrow \infty} \tau_{kl}(m+r) = 1/L(w^*). \quad (10.3.12)$$

由于 X_n 的第二项满足 $\hat{w}(n) = w^* (n > m)$, 故 $\lim_{n \rightarrow \infty} X_n = (\mathcal{I}[w^*], w^*)$ 成立.

(3) 特别地, 如果过程进入集合 $S_{|A|} \times \{w^*\}$, 则从式 (10.3.11)、(10.3.12) 和转移概率 p_{kl} 的定义得 $p_{kl}(n) \rightarrow 1$ 对任意 $(k, l) \in w^*$ 成立. 对所有的 $(k, l) \in w^*$ 作乘积运算, 产生的 $P_{w^*}(n)$ 为 1. 满足以上条件的最优路径 w^* 存在的概率为 1, 因此定理的第二个结论成立. ■

推论 10.3.1 在算法 GBAS/tdev 中, 设

$$\rho_n = \frac{c_n}{n \log(n+1)} \quad (n \geq 1), \quad \text{其中 } 0 < \lim_{n \rightarrow \infty} c_n < 1,$$

则定理 10.3.1 的结论成立.

证明 只要证明该条件与条件 (10.3.3)、(10.3.4) 等价即可. 当 n 足够大时有

$$1 - \frac{\log n}{\log(n+1)} = \frac{\log\left(1 + \frac{1}{n}\right)}{\log(n+1)} \sim \frac{1}{n \log(n+1)},$$

此处的 $a_n \sim b_n (n \rightarrow \infty)$ 意味着 $a_n/b_n \rightarrow 1 (n \rightarrow \infty)$. 由 $c_n \rightarrow c \in [0, 1]$, 对足够大的 n 有

$$\frac{c_n}{n \log(n+1)} < 1 - \frac{\log n}{\log(n+1)}.$$

另一方面, 由

$$\sum_{n=1}^{\infty} \frac{c_n}{n \log(n+1)} \geq \int_1^{\infty} \frac{dx}{x \log(x+1)} = \infty,$$

因此当 N 足够大时,

$$\sum_{n=1}^{\infty} \frac{c_n}{n \log(n+1)} \geq \sum_{n=N}^{\infty} \frac{c/2}{n \log(n+1)}$$

发散, 证明了条件 (10.3.4). ■

定理 10.3.2 在算法 GBAS/tdlb 中, 如果

$$\tau_{\min}(n) = \frac{c_n}{\log(n+1)} \quad (n \geq 1),$$

其中 $\lim_{n \rightarrow \infty} c_n > 0$, 则定理 10.3.1 成立 (证明同定理 10.3.1, 略).

三、一类收敛的蚁群优化算法^[13]

(一) 问题的表示与算法的描述

考虑极小化问题 (S, f, Ω) , 其中 S 为候选解集合, f 为目标函数, $f(s)$ 为候选解 $s \in S$ 的目标函数值, Ω 为约束条件集合, 通过 Ω 定义了可行候选解集合. 极小化问题的目的是找到一个最优的可行解 s^* , 使得 $f(s^*)$ 最小.

可将组合优化问题 (S, f, Ω) 映射到具有如下特征的问题:

- (1) 一个有限元素集 $C = \{c_1, c_2, \dots, c_{N_C}\}$;
- (2) X 为问题状态构成的有限集, 其元素由集合 C 中所有可能的序列 $x = \langle c_i, c_j, \dots, c_k, \dots \rangle$ 构成, $|x|$ 为序列 x 的长度 (即 x 包含的元素个数). 显然, 序列的最大长度 n 为一个有限整数;
- (3) 候选解 S 为 X 的一个子集, 即 $S \subseteq X$;
- (4) 可行状态集 \tilde{X} 为 X 的子集, 其定义与问题相关, 主要通过约束条件使序列 x 最终形成候选解;
- (5) S^* 为非空的最优解集合, 且 $S^* \subseteq \tilde{X}, S^* \subseteq S$.

有了以上的规范后, 人工蚂蚁在完全连接的带权图 $G = (C, L, T)$ 上随机行走以生成候选解. 在图 $G = (C, L, T)$ 中, 顶点为集合 C 中的元素, L 为 C 中所有顶点之间的连接集, T 是信息素矩阵. 图 G 称为构造图.

每次迭代开始, 蚂蚁随机地选择图 G 中的一个初始顶点, 然后根据剩余弧上的信息素强度逐步地随机选择下一个顶点, 直到生成问题的可行解为止. 其间, 约束条件用来避免蚂蚁生成无效的解. 一旦所有蚂蚁都生成可行解之后, 将对信息素的迹进行更新. 一般蚂蚁构造解的过程如下:

procedure ANT.SOLUTION.CONSTRUCTION

while ($x_k \in \tilde{X}$ 且 $x_k \notin S$) **do**

在第 k 步生成序列 $x_k = \langle c_{i_1}, c_{i_2}, \dots, c_{i_k} \rangle$ 之后, 按式 (10.3.13) 随机的选择下一个顶点 $c_{i_{k+1}}$.

$$P(c_{i_{k+1}} = c | T, c_{i_k}) = \begin{cases} \frac{\tau(c_{i_k}, c)^\alpha}{\sum_{y \in C(c_{i_k}, y \in J_{c_{i_k}})} \tau(c_{i_k}, y)^\alpha}, & \text{如果 } (c_{i_k}, c) \in J_{c_{i_k}}, \\ 0, & \text{否则,} \end{cases} \quad (10.3.13)$$

end

end

其中 $0 < \alpha < +\infty$ 是一个参数, $J_{c_{i_k}}$ 为弧 (c_{i_k}, y) 的集合, 其中 $y \in C$ 且使得 $x_{i_{k+1}} = \langle c_{i_1}, c_{i_2}, \dots, c_{i_k}, y \rangle \in \tilde{X}$.

蚂蚁在前进过程中, 如果发现 $J_{c_{i_k}}$ 为空, 则构造解的过程结束. 当所有的蚂蚁结束解的构造过程, 则信息素的迹将会更新. 设 \hat{s} 为算法当前发现的最好解, s_t 为算法在第 t 次迭代时发现的最好解, $f(\hat{s}), f(s_t)$ 分别是相应的目标函数值. 所有弧上的信息素首先要蒸发一部分, 蒸发因子为 ρ ($0 < \rho < 1$). 然后对属于最优路径 \hat{s} 上的信息素进行增加. 信息素更新过程描述如下:

procedure PHEROMONE_UPDATE

$$\forall (i, j) : \tau(i, j) \leftarrow (1 - \rho) \cdot \tau(i, j) \quad .$$

$$\text{if } f(s_t) < f(\hat{s}) \text{ then } \hat{s} \leftarrow s_t$$

$$\forall (i, j) \in \hat{s} : \tau(i, j) \leftarrow \tau(i, j) + g(\hat{s})$$

$$\forall (i, j) : \tau(i, j) \leftarrow \max\{\tau_{\min}, \tau(i, j)\}$$

end

其中, ρ ($0 < \rho < 1$) 为蒸发系数, $\tau_{\min} > 0$ 是一个参数, $g(s)$ 为可行解集 S 到正实数上的一个函数, 即 $g: S \mapsto \mathbb{R}^+$, 且 $0 < g(s) < +\infty, f(s) < f(s') \Rightarrow g(s) \geq g(s')$.

算法的初始化过程如下:

procedure INITIALIZE

生成一个可行解 s' 且设置 $\hat{s} = s'$;

$\forall (i, j)$ 设置 $\tau(i, j) = \tau_0$;

for 每只蚂蚁 **do**

根据问题的相关条件, 选择一个起始定点 c_1 ;

设置 $k = 1$ 且 $x_k = \langle c_1 \rangle$.

end for

end

其中 τ_0 ($\tau_{\min} \leq \tau_0 < +\infty$) 是一个参数.

该算法称为 $ACO_{gb, \tau_{\min}}$, 其中的 gb 表示使用全局最优信息素更新规则. 算法首先由 INITIALIZE 过程初始化, 然后由 ANT.SOLUTION.CONSTRUCTION 和 PHEROMONE.UPDATE 过程不断迭代, 直到满足终止条件. 算法中假设 $\tau_{\min} < g(s^*)$, 这可通过设置 $\tau_0 = g(s')/2$ 办到.

(二) 算法的收敛性结论及证明

引理 10.3.1 对任意的 τ_{ij} , 有下面的不等式成立:

$$\lim_{t \rightarrow \infty} \tau_{ij} \leq \tau_{\max} = \frac{1}{\rho} \cdot g(s^*). \quad (10.3.14)$$

证明 在每次迭代时, 弧 (i, j) 上增加的信息素不会超过 $g(s^*)$. 特别地, 第 1 次迭代后最大可能的信息素为 $(1 - \rho) \cdot \tau_0 + g(s^*)$, 第 2 次为 $(1 - \rho)^2 \cdot \tau_0 + (1 - \rho)g(s^*) + g(s^*)$, \dots , 第 t 次迭代后信息素的最大取值为

$$\tau_{ij}^{\max}(t) = (1 - \rho)^t \cdot \tau_0 + \sum_{i=1}^t (1 - \rho)^{t-i} \cdot g(s^*). \quad (10.3.15)$$

由 $0 < \rho < 1$, 考虑渐近的情况, 其和将收敛到

$$\tau_{\max} = \frac{1}{\rho} \cdot g(s^*). \quad \blacksquare$$

引理 10.3.2 在发现最优路径之后, 由于使用全局最优信息素更新规则且 $\tau_{ij}^*(t) \geq \tau_{\min}$, 因此 $\tau_{ij}^*(t)$ 单调递增, 且有

$$\forall (i, j) \in s^* : \lim_{t \rightarrow \infty} \tau_{ij}^*(t) = \tau_{\max} = \frac{1}{\rho} \cdot g(s^*),$$

其中, τ_{ij}^* 是弧 $(i, j) \in s^*$ 上的信息素 (证明从略).

定理 10.3.3 设 $P^*(t)$ 为在第 t 次迭代时, 算法首次发现最优解的概率, 则对任意小 $\varepsilon > 0$ 和足够大的迭代次数 t , 有

$$P^*(t) \geq 1 - \varepsilon,$$

且

$$\lim_{t \rightarrow \infty} P^*(t) = 1.$$

证明 由于信息素的值被限制在 τ_{\min} 和 τ_{\max} 之间, 因此式 (10.3.13) 的可行选择 $p_{\min} > 0$, p_{\min} 的一个下界为

$$p_{\min} \geq \hat{p}_{\min} = \frac{\tau_{\min}^{\alpha}}{(N_c - 1) \cdot \tau_{\max}^{\alpha} + \tau_{\max}^{\alpha}}, \quad (10.3.16)$$

其中 N_c 为集合 C 中元素个数, 则任意解 s' (包括任何最优解 $s^* \in S^*$) 可以以概率 $\hat{p} \geq \hat{p}_{\min}^n > 0$ 生成, 其中 n 为序列的最大长度. 因此 $P^*(t)$ 的一个下界为

$$P^*(t) = 1 - (1 - \hat{p})^t,$$

当选择足够大的 t 时, 概率 $P^*(t)$ 可以大于 $1 - \varepsilon$, 因此有 $\lim_{t \rightarrow \infty} P^*(t) = 1$.

定理 10.3.4 设 t^* 为首次发现最优解的迭代次数, 则存在一个 t_0 , 使得

$$\tau_{ij}(t) > \tau_{kl}(t),$$

$\forall (i, j) \in s^*, \forall (k, l) \in L \wedge (k, l) \notin s^*$ 成立, 其中 $\forall t > t^* + t_0 = t^* + \lceil (1 - \rho) / \rho \rceil$.

证明 由于算法中使用全局最优更新规则, 则当 $t > t^*$ 时, 只有最优路径上的信息素得到加强, 而其他边上的信息素由于蒸发而逐渐减少. 因此在有限次迭代之后, 会出现最优解路径上信息素的迹大于其他边上的. 下面讨论 t_0 的估计值.

设在第 t^* 次迭代后, 最优路径 s^* 的弧 (i, j) 上的信息素为 $\tau_{ij}^*(t^*) = \tau_{\min}$ (考虑最坏的情况), 另外, 设非最优路径上弧 (k, l) 的信息素为 $\tau_{kl}(t^*) = \tau_{\max}$. 在第 $t^* + t'$ 次迭代后, $\tau_{ij}^*(t)$ 为

$$\tau_{ij}^*(t^* + t') = (1 - \rho)^{t'} \cdot \tau_{\min} + \sum_{i=0}^{t'-1} (1 - \rho)^i \cdot g(s^*) > t' \cdot (1 - \rho)^{(t'-1)} \cdot g(s^*). \quad (10.3.17)$$

另外, 在第 $t^* + t'$ 次迭代后,

$$\tau_{kl}(t^* + t') = \max \left\{ \tau_{\min}, (1 - \rho)^{t'} \cdot \tau_{\max} \right\}.$$

欲使 $\tau_{ij}^*(t^* + t') > \tau_{kl}(t^* + t')$, 只需

$$t' \cdot (1 - \rho)^{(t'-1)} \cdot g(s^*) > (1 - \rho)^{t'} \cdot \tau_{\max},$$

即只需

$$t' > \left\lceil \frac{\tau_{\max} \cdot (1 - \rho)}{g(s^*)} \right\rceil = \left\lceil \frac{(1 - \rho)}{\rho} \right\rceil \equiv t_0. \quad \blacksquare$$

从定理 10.3.4 可知, 当 $t > t^* + t_0$ 时, 每只蚂蚁都能以确定的概率构造最优解 s^* .

定理 10.3.5 一旦发现最优路径后, 对任意 $\tau_{ij}(t)$, 其中 $(i, j) \notin s^*$, 有

$$\lim_{t \rightarrow \infty} \tau_{ij}(t) = \tau_{\min}.$$

证明 在第 $t^* + t'$ 次迭代时, 非最优路径上的信息素 $\tau_{ij}(t + t') = \max \{ \tau_{\min}, (1 - \rho)^{t'} \cdot \tau_{\max} \}$, 则当 $t \rightarrow \infty$ 时, $\tau_{ij}(t) \rightarrow \tau_{\min}$. ■

定理 10.3.6 从迭代次数 $t' (t' \geq t^* + t_0)$ 开始, 有 $\forall (i, j) \notin s^*, \tau_{ij}(t) = \tau_{\min}$. 其中, t^* 为首次发现最优解 s^* 的迭代次数, $t_0 = \lceil (\ln \tau_{\min} - \ln \tau_{\max}) / \ln(1 - \rho) \rceil$.

证明 由定理 10.3.5 知 $\tau_{ij}(t + t') = \max \{ \tau_{\min}, (1 - \rho)^{t'} \cdot \tau_{\max} \}$, 设 t_0 为满足不等式 $(1 - \rho)^{t_0} \cdot \tau_{\max} < \tau_{\min}$, 容易求得 $t_0 = \lceil (\ln \tau_{\min} - \ln \tau_{\max}) / \ln(1 - \rho) \rceil$. ■

推论 10.3.2 设 t^* 为首次发现最优解的迭代次数, $P(s^*, t, k)$ 为任意蚂蚁 k 在第 t 次迭代时经过最优路径 s^* 的概率, 当 $t > t^*$ 时, 有

$$\lim_{t \rightarrow \infty} P(s^*, t, k) \geq 1 - \hat{\varepsilon}(\tau_{\min}, \tau_{\max}).$$

证明 设蚂蚁 k 位于顶点 i , 弧 (i, j) 属于最优路径 s^* , 蚂蚁 k 将从节点集 J_i 中选择下一个节点 j . 蚂蚁 k 做出正确选择的概率 $p_{ij}^*(t)$ 的一个下界 $\hat{p}_{ij}^*(t)$ 为

$$\hat{p}_{ij}^*(t) = \frac{(\tau_{ij}^*(t))^\alpha}{(\tau_{ij}^*(t))^\alpha + \sum_{(i,k) \notin s^*} (\tau_{ik}(t))^\alpha}.$$

由定理 10.3.5, 10.3.6 得

$$\hat{p}_{ij}^* = \lim_{t \rightarrow \infty} \hat{p}_{ij}^*(t) = \frac{\lim_{t \rightarrow \infty} (\tau_{ij}^*(t))^\alpha}{\lim_{t \rightarrow \infty} \left[(\tau_{ij}^*(t))^\alpha + \sum_{(i,k) \notin s^*} (\tau_{ik}(t))^\alpha \right]} = \frac{\tau_{\max}^\alpha}{\tau_{\max}^\alpha + (N_C - 1) \cdot \tau_{\min}^\alpha}.$$

因此, $P(s^*, t, k)$ 的一个下界为 $\hat{p}_k^* = (\hat{p}_{ij}^*)^n$, 设置 $\varepsilon = 1 - \hat{p}_k^*$ 即可得结论. ■

第四节 本章小结

本章就 ACO 的理论部分进行了论述. ACO-MH 作为一种通用的优化技术, 在设计具体的算法之前, 首先要将待求解问题表示成相应的构造图的形式. 阐述了如何将待求解问题表示成相应的构造图, 并在此基础上讨论了 ACO-MH 的具体实现, 同时对 ACO-MH 中解的评估、蚂蚁数和信息素等的影响进行了分析. 在 ACO 的收敛性方面, 较详细论述了两类特殊 ACO-MH 算法的收敛性结果及其证明过程. 本章的内容作为 ACO 的理论基础, 对 ACO 算法的设计具有极强的指导意义.

参 考 文 献

- [1] Dorigo M, Di Caro G, and Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, 5(2): 137~172
- [2] Holland J H. *Adaptation in Nature and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975

- [3] 陈国良, 王煦法, 庄镇泉, 王东生. 遗传算法及其应用. 北京: 人民邮电出版社, 2001. 433
- [4] Dorigo M and Di Caro G. The Ant Colony Optimization meta-heuristic. In Corne D, Dorigo M, and Glover F, editors. *New Ideas in Optimization*. London, UK: McGraw Hill, 1999. 11~32
- [5] Gutjahr W J. A generalized convergence result for the graph-based ant system metaheuristic. Dept. Statistics and Decision Support Systems, Univ.Vienna, Austria, Tech. Rep. 99-09, 1999
- [6] Gutjahr W J. A graph-based ant system and its convergence. *Future Gener.Comput. Syst.*, 2000, 16(8): 873~888
- [7] Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. *Info. Processing Lett.*, 2002, 82(3): 145~153
- [8] Stützle T and Dorigo M. A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(4): 358~365
- [9] Meuleau N and Dorigo M. Ant colony optimization and stochastic gradient descent. *Artif. Life*, 2002, 8(2): 103~121
- [10] Di Caro G. and Dorigo M. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 1998, 9: 317~365
- [11] Di Caro G and Dorigo M. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Universit Libre de Bruxelles, Belgium, 1997
- [12] Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. *Info. Processing Lett.*, 2002, 82(3): 145~153
- [13] Stützle T and Dorigo M. A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(4): 358~365

第十一章 基本蚁群优化算法及其改进算法

第一节 引言

蚁群在觅食过程中总能找到蚁巢和食物源之间的最短路径. 受其启发, 意大利学者 M Dorigo, V Maniezzo 和 A Colomi^[1~3] 提出了一种新型的模拟进化算法——蚂蚁系统 (ant system, 简称 AS). AS 算法是第一个 ACO 算法, 被称为基本 ACO 算法, 该算法的出现, 开创了 ACO 研究的先河. AS 算法首先用来求解 TSP, 并获得了极大的成功. 实验结果显示 AS 算法具有极强的发现较好解的能力, 但同时也存在一些缺陷如收敛速度慢、易出现停滞现象等. 针对 AS 算法的不足, 许多学者对其进行了深入的研究, 提出了一些改进的 ACO 算法如最优保留蚂蚁系统 (Ant System with Elitist, 简称 AS_{elite})^[4]、蚁群系统 (Ant Colony System, 简称 ACS)^[5]、最大-最小蚂蚁系统 (max-min ant system, 简称 MMAS)^[6~9]、基于排序的蚂蚁系统 (rank-based version of ant system, 简称 AS_{rank})^[10] 等等.

本章首先介绍了 AS 算法的模型、实现及相关属性, 然后讨论了几种改进的 ACO 算法, 最后详细介绍了作者提出的两种改进的 ACO 算法及其实验结果.

第二节 蚂蚁系统及其属性

一、蚂蚁系统的模型与实现

(一) 旅行商问题

一般地, 旅行商问题 (traveling salesman problem, 简称 TSP) 可描述如下:

设 $C = \{c_1, c_2, \dots, c_n\}$ 为 n 个城市的集合, $L = \{l_{ij} | c_i, c_j \in C\}$ 是 C 中元素两两连接的集合, $G = (C, L)$ 是一个图, TSP 的目的是从 G 中找出长度最短的 Hamiltonian 圈, 即找出对 $C = \{c_1, c_2, \dots, c_n\}$ 中 n 个城市访问且只访问一次的最短的一条封闭曲线. TSP 分为对称型和非对称型. 在对称型 TSP 中, 有 $d_{ij} = d_{ji}, \forall c_i, c_j \in C(1, 2, \dots, n)$, d_{ij} 是 l_{ij} 的长度; 而在非对称型 TSP 中, 至少存在一对 $c_i, c_j \in C$, 使 $d_{ij} \neq d_{ji}$.

(二) 蚂蚁系统

设 $b_i(t) (i = 1, 2, \dots, n)$ 为 t 时刻城市 i 的蚂蚁数, 则 $m = \sum_{i=1}^n b_i(t)$ 为全部蚂蚁

数. 每只人工蚂蚁有以下特性:

(1) 蚂蚁根据某一概率函数选择下一座城市, 其中概率函数是城市间距离及连接边上信息素量的函数 (设 $\tau_{ij}(t)$ 为 t 时刻连接边 $e(i, j)$ 上的信息素量).

(2) 每只人工蚂蚁只能走合法路线, 除非一次周游 (蚂蚁走完所有的城市称为一次周游) 结束, 不允许转到已访问的城市. 该过程由蚂蚁的禁忌表来控制. 设 $tabu_k$ 为蚂蚁 k 的禁忌表, 则蚂蚁 k 在经过城市 i 以后, 就将城市 i 加入到自己的禁忌表 $tabu_k$ 中, 表示下一次不能再选择城市 i . 用 $tabu_k(s)$ 表示禁忌表中第 s 个元素, 也即蚂蚁所走过的第 s 个城市.

(3) 完成一次周游后, 蚂蚁在其访问过的每一条边上留下相应的信息素.

AS 算法可以表述如下: 在算法的初始时刻, 将 m 只蚂蚁随机地放到 n 座城市, 同时, 将每只蚂蚁的禁忌表的第一个元素设置为它当前所在的城市. 此时各路径上的信息素量相等, 设 $\tau_{ij}(0) = C$ (C 为一较小的常数). 接下来, 每只蚂蚁根据路径上残留的信息素量和启发式信息 (两城市间的距离) 独立地选择下一座城市. 在时刻 t , 蚂蚁 k 从城市 i 转移到城市 j 的概率 $p_{ij}^k(t)$ 为

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{s \in J_k(i)} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}(t)]^\beta}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则,} \end{cases} \quad (11.2.1)$$

其中, $J_k(i) = \{1, 2, \dots, n\} - tabu_k$ 表示蚂蚁 k 下一步允许选择的的城市集合. 列表 $tabu_k$ 记录了蚂蚁 k 当前走过的城市. 当所有 n 座城市都加入到 $tabu_k$ 中时, 蚂蚁 k 便完成了一次周游, 此时蚂蚁 k 所走过的路径便是 TSP 的一个可行解. 式 (11.2.1) 中的 η_{ij} 是一个启发式因子, 表示蚂蚁从城市 i 转移到城市 j 的期望程度. 在 AS 算法中, η_{ij} 通常取城市 i 与城市 j 之间距离的倒数. α 和 β 分别表示信息素和启发式因子的相对重要程度. 当所有蚂蚁完成一次周游后, 各路径上的信息素根据式 (11.2.2) 更新.

$$\tau_{ij}(t+n) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}, \quad (11.2.2)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (11.2.3)$$

其中, ρ ($0 < \rho < 1$) 表示路径上信息素的蒸发系数, $1-\rho$ 表示信息素的持久性系数, $\Delta\tau_{ij}$ 表示本次迭代边 ij 上信息素的增量. $\Delta\tau_{ij}^k$ 表示第 k 只蚂蚁在本次迭代中留在边 ij 上的信息素量. 如果蚂蚁 k 没有经过边 ij , 则 $\Delta\tau_{ij}^k$ 的值为零. $\Delta\tau_{ij}^k$ 表示为

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{若蚂蚁 } k \text{ 在本次周游中经过边 } ij, \\ 0, & \text{否则,} \end{cases} \quad (11.2.4)$$

其中, Q 为正常数, L_k 表示第 k 只蚂蚁在本次周游中所走过路径的长度.

AS 算法的伪代码如下:

Step 1 初始化

置 $t := 0$; $\{t$ 为计时器 $\}$

置 $NC := 0$; $\{NC$ 为迭代计数器 $\}$

对每条边 $e(i, j)$ 设置 $\tau_{ij}(t) = C, \Delta\tau_{ij}(t) = 0$; 将 m 只蚂蚁放到 n 座城市上.

Step 2 置 $s := 1$; $\{s$ 为禁忌表的索引 $\}$

for $k := 1$ to m do

将蚂蚁 k 的起点城市加入其禁忌表 $tabu_k$.

end for

Step 3 repeat until 禁忌表 $tabu_k$ 已满

设置 $s := s + 1$;

for $k := 1$ to m do

按式 (11.2.1) 计算转移概率 $p_{ij}^k(t)$, 根据赌轮方式选择下一个要到的城市 j
 $\{$ 在时刻 t , 蚂蚁 k 在城市 $i = tabu_k(s - 1)\}$

蚂蚁 k 移到城市 j

将城市 j 加入 $tabu_k(s)$

end for

end repeat

Step 4 for $k := 1$ to m do

蚂蚁 k 从 $tabu_k(n)$ 移到 $tabu_k(1)$;

计算蚂蚁 k 走过的周游长度 L_k ;

更新当前的最优路径.

end for

for 每条边 $e(i, j)$

for $k := 1$ to m do

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{如果 } e(i, j) \in tabu_k, \\ 0, & \text{否则;} \end{cases}$$

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^k.$$

end for

end for

Step 5 for 每条边 $e(i, j)$, 按式 (11.2.2) 计算 $\tau_{ij}(t + n)$;

置 $t := t + n$;

置 $NC := NC + 1$;

for 每条边 $e(i, j)$, 置 $\Delta\tau_{ij} := 0$.
Step 6 if($NC < NC_{MAX}$) and(没有出现停滞情况)then
 清空所有禁忌表;
 goto step 2
 else
 打印最优路径;
 算法停止.
 end if

M Dorigo 提出了 3 种 AS 算法的模型^[4], 算法 2.1 称为 ant-cycle; 另外两个模型分别称为 ant-quantity 和 ant-density, 其差别主要在式 (11.2.4), 即在 ant-quantity 模型中为

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{d_{ij}}, & \text{蚂蚁 } k \text{ 在时刻 } t \text{ 和 } t+1 \text{ 经过边 } ij, \\ 0, & \text{否则.} \end{cases} \quad (11.2.5)$$

在 ant-density 模型中为

$$\Delta\tau_{ij}^k = \begin{cases} Q, & \text{若蚂蚁 } k \text{ 在时刻 } t \text{ 和 } t+1 \text{ 经过边 } ij, \\ 0, & \text{否则.} \end{cases} \quad (11.2.6)$$

AS 算法实际上是正反馈原理和启发式算法相结合的一种算法. 在选择路径时, 蚂蚁不仅利用了路径上的信息素, 而且用到了城市间距离的倒数作为启发式因子. 实验结果表明, ant-cycle 模型比 ant-quantity 和 ant-density 模型有更好的性能. 这是因为 ant-cycle 模型利用全局信息更新路径上的信息素量, 而 ant-quantity 和 ant-density 模型使用局部信息. AS 算法的时间复杂度为 $O(NC \cdot n^3)$, 其中 NC 表示迭代的次数, n 为城市数. 在文献 [4] 中, M Dorigo 通过求解 Oliver30(30 城市的 TSP) 问题对 AS 算法的参数进行了分析, 发现当 $\alpha = \{0.5, 1\}$, $\beta = \{1, 2, 3, 4, 5\}$ 时, AS 算法总能收敛到当前最优解, 当蚂蚁数 m 接近城市数 n 时, 算法有较好的性能.

二、蚂蚁系统的参数设置和基本属性

目前, 对 AS 算法的参数设置和属性的研究大多还处于实验阶段. M Dorigo^[1,4] 等人通过大量的实验对蚂蚁系统的参数和基本属性进行了探讨. 讨论的参数包括:

- α ——信息素的相对重要程度;
- β ——启发式因子的相对重要程度;
- ρ ——信息素蒸发系数 ($(1 - \rho)$ 表示信息素的持久性系数);
- Q ——蚂蚁释放的信息素量.

在实验中,为了观察某个参数对算法性能的影响,在测试该参数时,其他参数取缺省值。各参数的缺省值为 $\alpha = 1, \beta = 1, \rho = 0.5, Q = 100$ 。

为比较三种模型的性能, M Dorigo 首先通过实验方法确定每种模型的最优参数集,然后各模型在取最优参数集的情况下,分别求解 Oliver30 问题十次,其结果如表 11-1 所示(实验发现参数 Q 对 AS 算法性能的影响不大,故表中没有列出)。

表 11-1 模型 ant-density, ant-quantity, ant-cycle 的比较 (M Dorigo,1996)

模型	参数集		
	最好参数集	平均结果	最好结果
ant-density	$\alpha = 1, \beta = 5, \rho = 0.01$	426.740	424.635
ant-quantity	$\alpha = 1, \beta = 5, \rho = 0.01$	427.315	426.255
ant-cycle	$\alpha = 1, \beta = 5, \rho = 0.5$	424.250	423.741

从表 11-1 可以看出, ant-cycle 模型的性能要优于 ant-density 和 ant-quantity 模型。原因是在 ant-cycle 模型中用到了全局信息,即蚂蚁释放在路径上的信息素量与其所得解的质量成正比。周游长度越短的蚂蚁,释放在其经过的路径上的信息素量就越多,而 ant-density、ant-quantity 模型在搜索解时,只使用了局部信息,没有用到任何解的信息。

在 ant-cycle 算法中, $\rho = 0.5$ 可以解释为:在计算的初期,使用启发式贪婪搜索策略;后期,主要使用路径上的信息素 τ_{ij} 来进行搜索,同时, ant-cycle 能够忘记部分过去的经历,以便更好的利用新的全局信息来搜索较好的解。下面主要就 ant-cycle 模型来研究 AS 的特性。

(一) 信息素的分布

图 11-1 是 ant-cycle 模型求解 10 城市 TSP(CCA0) 时各边上信息素分布的进化过程。图中边的长度与城市间的距离成正比,边的粗细与其上信息素量成正比。图 11-1(a) 显示初始时刻各边上信息素量相等,图 11-1(b) 为算法迭代 100 次

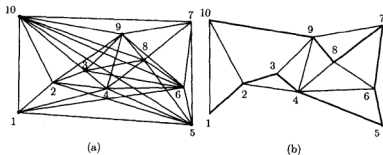


图 11-1 10 城市 TSP(CCA0) 信息素迹的进化过程

(a) 初始时刻信息素迹的分布; (b) 算法迭代 100 次以后信息素迹的分布。

后, 较好解路径上的信息素量明显高于其他边上的. 由于信息素蒸发的影响, 较差解路径上的信息素量将越来越少. 可见, 在算法迭代一定次数后, 蚂蚁将搜索空间缩小为可能存在最优解的一个子空间, 这样可以极大地提高蚂蚁搜索最优解的效率, 但同时也容易引起停滞现象, 使算法陷入局部最优.

(二) 参数 α 、 β 对 AS 算法性能的影响

定义 11.2.1 在蚂蚁搜索解的过程中, 所有蚂蚁都选择同样的路径, 即系统不再搜索较好的解, 称为停滞现象 (Stagnation behavior).

当参数设置为某些值时, 算法迭代到一定代数后将出现停滞现象. 其原因是因为较好路径上的信息素远大于其他边上的, 从而使所有蚂蚁都选择相同的路径.

定义 11.2.2 设 $\tau_{\min}(r, s)$ 、 $\tau_{\max}(r, s)$ 分别为与节点 r 相连的边上最大、最小信息素值, 令 $\delta(r) = \tau_{\max}(r, s) - \tau_{\min}(r, s)$, 对某个给定的 $\lambda (0 < \lambda < 1)$, 则在所有与节点 r 相连的边中, 信息素量大于等于 $\lambda \cdot \delta(r) + \tau_{\min}(r, s)$ 的边的数量即为节点 r 的节点分支数 (node branching). 其中 λ 可根据实际需要确定.

定义 11.2.3 设 $\theta(r)$ 为节点 $r (r = 1, 2, \dots, n)$ 的节点分支数, n 为节点数, 则平均节点分支数 (average node branching, 简称 ANB) 为

$$\sum_{r=1}^n \theta(r) / n.$$

图 11-2 是 ant-cycle 求解 Oliver30 问题时 ANB 的进化情况. 在某些参数设置下, 当算法迭代 2500 次后, ANB 到达 2. 就对称 TSP 而言, 这意味着所有的蚂蚁都选择同样的路径, 即算法出现停滞现象.

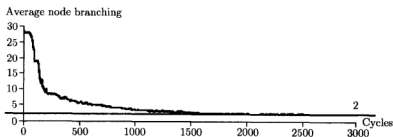
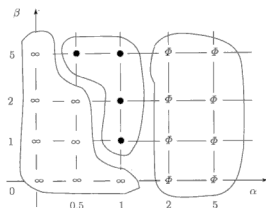


图 11-2 在 $\alpha = 5, \beta = 2$ 时平均节点分支数进化到 2, 出现停滞现象 (M Dorigo, 1996)

图 11-3 是 ant-cycle 模型中参数 α 、 β 对算法性能的影响. 此图中的三个区域解释如下:

(1) 算法不能发现最好解, 且出现停滞现象. 图中 Φ 表示的区域. 对于较高的 α 值, 该算法很快出现停滞现象且不能发现较好的解;

(2) 算法不能发现最好解, 且不出现停滞现象. 图中 ∞ 表示的区域. 将 α 设置为较小的值, 该算法很难找到较好的解;


 图 11-3 ant-cycle 模型中不同的参数 (α 、 β) 组合对算法的影响 (M Dorigo, 1996)

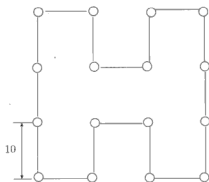
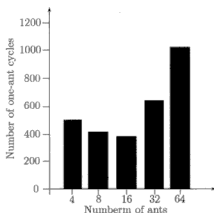
- 算法能发现已知的最好解, 且不出现停滞现象; ∞ — 算法不能发现最好解, 且不出现停滞现象;
- Φ — 算法不能发现最好解, 且出现停滞现象。

(3) 算法能发现已知的最好解, 不出现停滞现象。图中 ● 表示的区域。在这样的情况下, 不同的参数组合 ($\alpha = 1, \beta = 1$; $\alpha = 1, \beta = 2$; $\alpha = 1, \beta = 5$; $\alpha = 0.5, \beta = 5$) 算法具有同样的发现较好解的能力。即对于 Oliver30 问题, 在大致相同的迭代次数后都能找到最优解。

以上的实验结果可以解释如下: 当 α 取较大的值时, 意味着在选择路径时, 路径上的信息素非常重要; 当 α 取较小的值时, 则 ant-cycle 变成随机的贪婪算法。

(三) 蚂蚁数 m 对 AS 算法的影响

为了测试蚂蚁数 m 对 ant-cycle 模型性能的影响, 用该模型来求解图 11-4 所


 图 11-4 4×4 方格问题的一个可选解
(M Dorigo, 1996)

 图 11-5 在求解 4×4 网格问题时, 每只蚂蚁找到最优解的周游数与蚂蚁总数之间的关系, 5 次运行的平均结果 (M Dorigo, 1996)

示的 4×4 方格问题。众所周知, 当方格的边长为 10 时, 其最优解为 160。在该实验中, 分别取蚂蚁数为 $m \in \{4, 8, 16, 32, 64\}$ 。

图 11-5 为实验所得的结果, 其中横轴表示蚂蚁数, 纵轴表示发现最优解所用的迭代次数。从图中可以看出当 $M \approx N$ 时, ant-cycle 可以在最少的迭代次数内找到最优解。在对随机分布的 16 城市的实验中也得到同样的结果。

(四) 协同作用对 AS 算法的影响

如图 11-6 所示, 图 11-6(a) 为蚂蚁间没有通讯 ($\alpha=0$) 时的情形, 此时 AS 算法变成贪婪搜索算法; 图 11-6(b) 为有协同作用 ($\alpha=1$) 时的情形。从图中可以看出, 有协同作用时提高了 AS 算法的性能。

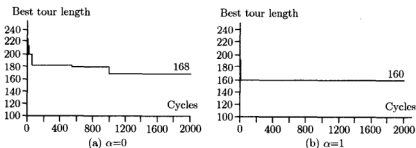


图 11-6 有协同工作 ($\alpha=1$) 与无协同工作 ($\alpha=0$) 相比, 提高了算法的性能 (M Dorigo, 1996)

(五) 蚂蚁的初始分布

为了测试蚂蚁的初始分布对 AS 算法性能的影响, M Dorigo^[4] 分别对随机分布的 16 城市的 TSP, 4×4 网格问题和 Oliver30 问题进行了测试。分两种情况: (i) 所有蚂蚁初始时刻放在同一个城市; (ii) 蚂蚁分布在不同的城市中。结果发现第 (ii) 种情况可获得较高的性能, 同时也测试了随机分布与统一分布的性能差异, 结果发现其差别不大。

第三节 改进的蚁群优化算法

一、蚂蚁系统的优点与不足

AS 算法具有如下优点:

- (1) 较强的鲁棒性。对 AS 算法的模型稍加修改, 便可以应用于其他问题;
- (2) 分布式计算。AS 算法是一种基于种群的进化算法, 本质上具有并行性, 易于并行实现;

(3) 易于与其他方法结合. AS 算法很容易与多种启发式算法结合, 以改善算法的性能.

众多研究已经证明 AS 算法具有很强的发现较好解的能力, 这是因为该算法不仅利用了正反馈原理, 在一定程度上可以加快进化过程, 而且是一种本质上并行的算法. 不同个体之间不断进行信息交流和传递, 从而能够相互协作, 有利于发现较好解. AS 算法可以解释为一种特殊的强化学习算法^[11], 公式 (11.2.1) 反映了 AS 算法与 Q 学习算法之间的联系, 相当于 Q 学习中的 Q 值, 表示学习所得到的经验. 虽然 AS 算法有许多优点, 但同时也存在一些缺陷, 如:

(1) 与其他方法相比, 该算法一般需要较长的搜索时间, AS 算法的复杂度可以反映这一点;

(2) 该方法容易出现停滞现象, 即搜索进行到一定程度后, 所有个体所发现的解完全一致, 不能对解空间进一步进行搜索, 不利于发现更好的解.

对于这两个问题, 已经引起了许多研究者的注意, 并提出了若干改进的蚂蚁算法, 如 M Dorigo 提出的蚁群系统 (ACS)^[12]、T Stützle^[6~9] 等人提出的最大-最小蚂蚁系统 (MMAS) 和 Bernd Bullnheimer^[10] 等人提出的基于排序的蚂蚁系统 (AS_{rank}) 等. 下面就这些有代表性的改进算法进行讨论.

二、最优解保留策略蚂蚁系统

通过使用最优蚂蚁可以提高蚂蚁系统中解的质量^[4]. 在最优解保留策略蚂蚁系统 (Ant System with Elitist, 简称 AS_{elite}) 中, 每次迭代完成后, 全局最优解得到更进一步的利用, 即在对信息素的迹进行更新时, 就好像有许多的最优蚂蚁选择了该路径. 与 AS 算法相比, AS_{elite} 算法在信息素更新时加强了对全局最优解的利用, 其信息素更新策略为

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^*, \quad \rho \in (0, 1), \quad (11.3.1)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (11.3.2)$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{如果蚂蚁 } k \text{ 经过边 } ij, \\ 0, & \text{否则,} \end{cases} \quad (11.3.3)$$

$$\Delta\tau_{ij}^* = \begin{cases} \sigma \cdot \frac{Q}{L^{gb}}, & \text{如果边 } (i, j) \text{ 是当前最优解的一部分,} \\ 0, & \text{否则,} \end{cases} \quad (11.3.4)$$

其中 $\Delta\tau_{ij}^*$ 为最优蚂蚁在边 (i, j) 上增加的信息素量, σ 为最优蚂蚁数, L^{gb} 为全局最优解.

图 11-7 为 AS_{elite} 算法求解 Oliver30 问题时算法达到局部最优解所用的迭代次数与最优蚂蚁数之间的关系 (实验的最大迭代次数为 2500)。实验结果显示最优蚂蚁数有一定的范围, 当最优蚂蚁数小于该范围时, 随着最优蚂蚁数的增加算法发现较好解的能力增加且缩短了发现较好解的时间; 但最优蚂蚁数超过该范围时, 算法的性能会随着最优蚂蚁数的增加而降低。

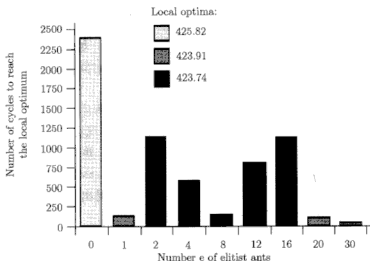


图 11-7 算法达到局部最优所需的迭代次数与最优蚂蚁数之间的关系, 5 次运行的平均结果 (M Dorigo, 1996)

三、蚁群系统

蚁群系统 (Ant Colony System, 简称 ACS) 是 AS 算法的改进版本, 与 AS 算法的主要区别在于: (i) 在选择下一座城市时, ACS 算法更多地利用了当前的较好解; (ii) 只在全局最优解所属的边上增加信息素; (iii) 每次当蚂蚁从城市 i 转移到城市 j 时, 边 ij 上的信息素将会适当的减少。

(一) 可行解的构造

在 ACS 算法中, 蚂蚁使用伪随机比率选择规则 (pseudo random proportional action choice rule) 选择下一座城市。即对位于城市 i 的蚂蚁 k , 以概率 q_0 移动到城市 l , 其中 l 为使 $\tau_{il}(t) \cdot [\eta_{il}]^\beta$ 达到最大的城市。该选择方式意味着蚂蚁将以概率 q_0 将最大可能的城市选入蚂蚁所构造的解; 除此之外, 蚂蚁以 $(1 - q_0)$ 的概率按式 (11.3.6) 选择下一座城市 j 。在 ACS 算法中, 蚂蚁的状态转移公式为

$$j = \begin{cases} \arg \max_{u \in allowed_k} [\tau_{iu}(t)] [\eta_{iu}]^\beta, & \text{如果 } q \leq q_0, \\ S, & \text{否则,} \end{cases} \quad (11.3.5)$$

其中 $q_0 \in (0, 1)$ 为常数, $q \in (0, 1)$ 为随机数, $\tau_{iu}(t)$ 表示 t 时刻城市 i 与城市 u 之间的信息素, η_{iu} 表示城市 i 与城市 u 之间的启发式因子, β 表示启发式因子的相对强弱. 在选择下一座城市之前随机生成 q , 如果 q 的值小于等于常数 q_0 , 则从城市 i 到所有可行的城市中找到 $[\tau_{iu}(t)] \cdot [\eta_{iu}]^\beta$ 最大的城市, 即为下一个要选择的城
市; 如果随机数 q 大于 q_0 , 则按式 (11.3.6) 来选择下一座城市.

$$p_k(i, j) = \begin{cases} \frac{[\tau_{ij}(t)] \cdot [\eta_{ij}]^\beta}{\sum_{s \in J_k(i)} [\tau_{is}(t)] \cdot [\eta_{is}]^\beta}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则,} \end{cases} \quad (11.3.6)$$

其中 $J_k(i)$ 为蚂蚁 k 当前的可行城市集合.

(二) 局部信息素更新

局部信息素更新的作用是使已选的边对后来的蚂蚁具有较小的影响力, 从而使蚂蚁对没有被选中的边有更强的探索能力. 在 ACS 算法中, 当蚂蚁从城市 i 转移到城市 j 后, 边 ij 上的信息素量按式 (11.3.7) 进行更新:

$$\tau_{ij} = (1 - \xi) * \tau_{ij} + \xi * \tau_0, \quad (11.3.7)$$

其中 τ_0 为常数, $\xi \in (0, 1)$ 为可调参数.

(三) 全局信息素更新

针对全局最优解所属的边按式 (11.3.8) 进行更新:

$$\tau_{ij}(t+1) = (1 - \rho) * \tau_{ij}(t) + \rho * \Delta\tau_{ij}^{gb}, \quad \rho \in (0, 1), \quad (11.3.8)$$

$$\Delta\tau_{ij}^{gb} = \begin{cases} 1/L^{gb}, & \text{如果边 } (i, j) \text{ 包含在全局最优路径中,} \\ 0, & \text{否则,} \end{cases} \quad (11.3.9)$$

其中 L^{gb} 为当前最好解的长度, ρ 为信息素蒸发系数.

M Dorigo^[12] 将 ACS 算法与 AS 算法, 模拟退火 (SA), 进化规划 (EP), 遗传算法 (GA), 模拟 - 遗传 (遗传算法与模拟退火算法的组合, 简称 AG) 进行了比较, 发现 ACS 算法在大多数情况下要优于其他算法, 至少性能相当. 在解决非对称 TSP 时, ACS 算法更具优势. 表 11-2 为对比实验结果 (表中显示了问题的整数解、小数解 (在圆括号中) 和发现最优整数解所需的迭代次数, 问题的最优解用黑体加粗显示).

表 11-2 ACS 与 GA, EP, SA, AG 的对比 (M Dorigo)

问题名称	ACS	GA	EP	SA	AG	最优解
Oliver30	420	421	420	424	420	420
(30 城市问题)	(423.74) [830]	(N/A) [3200]	(423.74) [40000]	(N/A) [24617]	(N/A) [12620]	(423.74)
Eil50	425	428	426	443	436	425
(50 城市问题)	(427.96) [1830]	(N/A) [25000]	(427.86) [100000]	(N/A) [68512]	(N/A) [28111]	(N/A)
Eil75	535	545	542	580	561	535
(75 城市问题)	(542.31) [3480]	(N/A) [80000]	(549.18) [325000]	(N/A) [173250]	(N/A) [95506]	(N/A)
KroA100	21282	21761	N/A	N/A	N/A	21282
(100 城市问题)	(21285.44) [4820]	(N/A) [103000]	(N/A) [N/A]	(N/A) [N/A]	(N/A) [N/A]	(N/A)

从表 11-2 中可以看出, ACS 算法和 EP 大大优于 GA, SA 和 AG.

四、最大 - 最小蚂蚁系统

最大 - 最小蚂蚁系统 (max-min ant system, 简称 MMAS)^[6~9] 是到目前为止解决 TSP, QAP 等问题最好的 ACO 算法. 和其他寻优算法比较起来, 它仍然属于最好的解决方案之一. MMAS 算法直接来源于 AS 算法, 主要作了如下的改进:

(i) 每次迭代结束后, 只有最优解所属路径上的信息被更新, 从而更好地利用了历史信息 (这与 ACS 算法的调整方案有点类似); (ii) 为了避免算法过早收敛于并非全局最优的解, 将各条路径可能的外激素浓度限制于 $[\tau_{\min}, \tau_{\max}]$, 超出这个范围的值被强制设为 τ_{\min} 或者是 τ_{\max} , 可以有效地避免某条路径上的信息量远大于其余路径, 使得所有的蚂蚁都集中到同一条路径上, 从而使算法不再扩散; (iii) 初始时刻, 各条路径上外激素的起始浓度设为 τ_{\max} , 在算法的初始时刻, ρ 取较小的值时, 算法有更好的发现较好解的能力. 所有蚂蚁完成一次迭代后, 按式 (11.3.10) 对路径上的信息作全局更新:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{best}(t), \quad \rho \in (0, 1), \quad (11.3.10)$$

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L^{best}}, & \text{如果边 } (i, j) \text{ 包含在最优路径中,} \\ 0, & \text{否则.} \end{cases} \quad (11.3.11)$$

允许更新的路径可以是全局最优解, 或本次迭代的最优解. 实践证明逐渐增加全局最优解的使用频率, 会使该算法获得较好的性能.

五、基于排序的蚂蚁系统

基于排序的蚂蚁系统 (rank-based version of ant system, 简称 AS_{rank}) 也是 AS 算法的改进版本. AS_{rank} 在每次迭代完成后, 蚂蚁所经路径将按从小到大的顺序排列, 即 $L^1(t) \leq L^2(t) \leq \dots \leq L^m(t)$, 并根据路径长度赋予不同的权重, 路径长度越短权重越大. 全局最优解的权重为 w , 第 r 个最优解的权重为 $\max\{0, w - r\}$, 按式 (11.3.12) 更新各路径上的信息素:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \sum_{r=1}^{w-1} (w - r) \cdot \Delta\tau_{ij}^r(t) + w \cdot \Delta\tau_{ij}^{gb}(t), \rho \in (0, 1), \quad (11.3.12)$$

其中, $\Delta\tau_{ij}^r(t) = 1/L^r(t)$, $\Delta\tau_{ij}^{gb}(t) = 1/L^{gb}$.

六、各种蚁群优化算法的比较

同 AS 算法相比, 以上算法的共同之处在于加强了对最优解的利用. 如在 ACS 算法和 MMAS 算法中, 只有最优解 (全局最优或本次迭代最优) 所属路径上的信息素允许增强. 在 AS_{rank} 算法中, 根据每次迭代路径的长短赋予不同的权重, 即对较短的路径赋予较大的权重. 这样最优解包含的路径将会有更多的机会被下一次选中. 但是, 加强对最优解的利用将会导致搜索中的停滞现象. 在 ACS 算法中通过增加局部信息素更新来减少路径上的信息素量, 从而使后面的蚂蚁选择该路径的可能性减少; 在 MMAS 算法中, 通过限制信息量的范围, 使路径上的信息量不会小于某一最小值, 从而避免了所有蚂蚁选择同一条路径的可能性, 即避免了搜索中的停滞现象.

表 11-3 是 MMAS, ACS, AS_{elite} 和 AS 算法在求解个对称 / 非对称 TSP 时的对比结果. 对于对称 TSP, 各算法迭代 10000 次, 对于非对称 TSP, 迭代 20000 次.

表 11-3 MMAS, ACS, AS_{elite} 与 AS 算法求解不同 TSP 实例的结果比较

TSP 实例	MMAS	ACS	AS_{elite}	AS
Eil51.tsp	427.6	428.1	428.3	437.3
kroA100.tsp	21320.3	21420.0	21522.83	22471.4
D198.tsp	15972.5	16054.0	16205.0	16705.6
Lin318	42220.2	42570.0	43422.8	45535.2
Ry48p.atasp	14553.2	14565.4	14685.2	15296.4
Ft70.atasp	39040.2	39099.0	39261.8	39596.3
Kro124p.atasp	36773.5	36857.0	37510.2	38733.1
Ft170.atasp	2828.8	2826.5	2952.4	3154.5

从表 11-3 可以看出, MMAS 算法具有最好的性能, 其次是 ACS 算法. 在通常情况下 AS_{elite} 算法优于 AS 算法.

图 11-8 是这四中算法在求解 TSP 时的典型收敛情况.

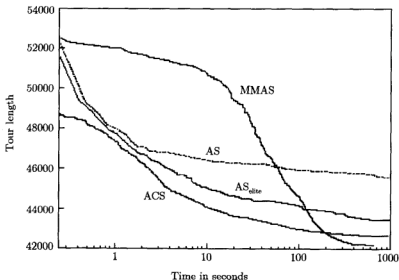


图 11-8 MMAS, ACS, AS_{elite} 与 AS 算法曲线收敛曲线的比较 (M Dorigo, 1999)

第四节 一种新的自适应蚁群算法^[14]

一、引言

AS 算法已经在许多领域证明是非常有效的算法^[14], 但却存在着收敛速度慢和容易出现停滞现象的缺陷^[4]. 文献^[15]对 AS 算法、演化计算、禁忌搜索、模拟退火、贪婪算法和局部搜索等元启发式算法进行了分析, 认为使这些算法保持高效的搜索能力同时又能够避免停滞现象的关键是在“探索”和“利用”(exploration & exploitation)之间建立一个平衡点, 也就是说既要使算法的搜索空间尽可能的大, 以寻找那些可能存在最优解的解区间; 同时又要充分利用群体内当前的有效信息, 使算法搜索的侧重点放在那些可能具有较高适应值的个体所在的区间内, 即缩小算法的搜索空间, 从而使算法在可以接受的时间内收敛到全局最优解. 下面针对 AS 算法的不足, 提出了一种自适应蚁群算法 (adaptive ant colony algorithm, 简称 AACA), 该算法根据平均节点分支数动态地调整转移概率以避免算法出现停滞现象, 从而极大地提高了算法搜索较好解的能力. 对 TSP 的仿真实验结果表明, AACA 算法即使在运行的后期, 仍然能以极大的概率搜索较好解.

二、AACA 算法设计

图 11-9 是 AS 算法求解 Oliver30 问题时平均节点分支数 (ANB) 的变化情况。算法在迭代 2500 次后, ANB 到达 2, 就对称 TSP 而言, 这意味着所有的蚂蚁都选择同样的路径, 即算法出现停滞现象。可见, ANB 可以作为衡量算法内部解的多样性的重要指标。

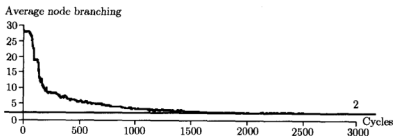


图 11-9 平均节点分支数进化到 2, 算法出现停滞现象 (M Dorigo, 1996)

与 AS 算法相比, AACA 算法作了如下几个方面的改进。

(一) 状态转移规则

由于 ANB 可以作为反映算法搜索能力的标志, 为了使蚁群始终能够在“探索”和“利用”之间保持平衡, 即使算法具有较强搜索能力的同时, 避免出现停滞现象, AACA 算法使用自适应伪随机比率选择规则 (adaptive pseudo random proportional action choice rule)。即对位于城市 i 的蚂蚁 k , 按式 (11.4.1) 选择下一座城市 j :

$$j = \begin{cases} \arg \max_{u \in J_k(i)} \{\tau(i, u) \cdot \eta^\beta(i, u)\}, & \text{如果 } q \leq q(\lambda(t)), \\ S, & \text{否则,} \end{cases} \quad (11.4.1)$$

$$q(\lambda(t)) = \lambda(t)/N, \quad (11.4.2)$$

其中 $\lambda(t) \in [2, N]$ 表示算法在第 t 次迭代时的 ANB, N 表示 TSP 中的城市数, 显然有 $q(\lambda(t)) \in [2/N, 1]$. q 为 $[0, 1]$ 上一致分布的随机数. $\tau(i, u)$ 表示城市 i 与城市 u 之间的信息素量, $\eta(i, u)$ 表示城市 i 与城市 u 之间的启发式因子, β 表示启发式因子的强弱。

蚂蚁在选择下一座城市之前随机生成 q , 如果 q 的值小于等于 $q(\lambda(t))$, 则从城市 i 到所有可行的城市中找出使 $\{\tau(i, u) \cdot \eta^\beta(i, u)\}$ 最大的城市, 即为下一个要到达

的城市; 如果随机数 q 大于 $q(\lambda(t))$, 则按式 (11.4.3) 选择下一座城市.

$$p_k(i, j) = \begin{cases} \frac{[\tau(i, j)] \cdot [\eta(i, j)]^\beta}{\sum_{s \in J_k(i)} [\tau(i, s)] \cdot [\eta(i, s)]^\beta}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则.} \end{cases} \quad (11.4.3)$$

从式 (11.4.2) 可以看出, 函数 $q(\lambda(t))$ 的值与平均节点分支数成正比. 当平均节点分支数 $\lambda(t)$ 变小时 (如接近 2), 说明算法将要出现停滞现象, 此时 $q(\lambda(t))$ 的值也会变小, 按照式 (11.4.3) 进行选择概率增大, 从而提高算法的随机搜索能力, 增大解的搜索空间; 当 ANB 变大时 (如接近城市数 N), 则按照式 (11.4.1) 进行选择的可能性增大, 即较好的路径将会被选中. 这样, AACA 算法能够根据自身的内部状态自适应地选择路径, 不但避免停滞现象, 而且保持较高的搜索能力.

(二) 局部信息素更新

蚂蚁从城市 i 转移到城市 j 后, 路径 ij 上的信息素按式 (11.4.4) 进行更新:

$$\tau_{ij} = (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_0, \quad (11.4.4)$$

其中 τ_0 为常数, $\xi \in (0, 1)$ 为可调参数.

(三) 全局信息素更新

针对全局最优解所属的边按式 (11.4.5) 进行信息素更新:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}^{gb}(t), \quad (11.4.5)$$

其中 $\Delta\tau_{ij}^{gb} = 1/L^{gb}$, L^{gb} 为当前全局最优解的长度, $\rho \in (0, 1)$ 为信息素的蒸发系数.

三、实验结果

实验选用 AS 算法, ACS 算法与 AACA 算法针对 51 座城市的 TSP eil51 进行对比, 三种算法的参数设置如表 11-4.

表 11-4 AS、ACS 和 AACA 算法的参数设置

AS 算法参数				ACS 算法参数			AACA 算法参数		
α	β	ρ	Q	β	ξ	ρ	β	ξ	ρ
1	5	0.50	100	2	0.95	0.95	2	0.9	0.9

三种算法中的蚂蚁数都为 40, 最大迭代次数为 2500, 10 次典型运行的结果如表 11-5 所示 (数字加粗表示较好解).

从实验结果可以看出, AACA 算法具有更强的发现较好解的能力. 10 次测试的平均结果也优于 AS 和 ACS 算法, 从统计的标准方差 Std. Dev. 来看, AACA 算法具有更好的稳定性.

图 11-10 表示在参数保持不变的情况下, 用 AACA 算法求解 eil51 时 ANB 的变化情况. 从图中可以看出, AACA 算法在搜索解的过程中, ANB 总是保持在的水平, 且处在不断的变化之中, 说明算法能够动态调节自己的搜索能力, 一方面可以使其具有较高的搜索较好解的能力, 同时也可以避免停滞现象的出现.

表 11-5 AS、ACS 和 AACA 求解 eil51 问题的结果比较 (10 次典型运行的结果)

序号	AS 算法		ACS 算法		AACA 算法	
1	429		428		428	
2	432		429		428	
3	431		427		427	
4	429		428		428	
5	431		427		426	
6	438		434		431	
7	432		427		427	
8	441		430		431	
9	435		426		426	
10	432		427		426	
统计结果	Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
	433	3.887	428.3	2.312	427.8	1.874

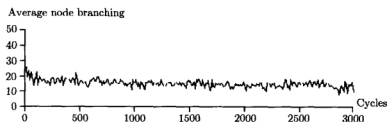


图 11-10 AACA 求解 eil51 问题时 ANB 的变化情况

第五节 基于混合行为的蚁群算法^[16]

一、引言

针对 AS 算法的缺点, 人们提出了许多改进的 ACO 算法^[5~9]. 文献 [6~9] 将信息素的值限制在 $[\tau_{\min}, \tau_{\max}]$ 内, 从而极大程度上避免了算法陷入局部最优; 文献 [17, 18] 只增加最优路径上信息素的浓度, 从而更好地利用了过去积累的知识, 以

加快算法的收敛速度;为避免算法出现停滞现象,文献 [13] 利用平均节点分支数作为衡量群体多样性的一种指标,并通过动态调整随机比例选择和确定性选择的比例来避免停滞现象;文献 [19] 根据算法搜索的情况动态调整蚂蚁释放的信息素量,即采用时变函数 $Q(t)$ 来代替 AS 算法中的常数 Q ;文献 [20] 提出一种自适应改变蒸发系数的策略来避免算法陷入停滞现象;文献 [21] 引入变异算子来避免算法出现停滞现象。

从以上的分析不难看出,改进的 ACO 算法主要从三个方面对 AS 算法作了修改: (i) 将信息素限制在一定区间内,避免某些边上的信息素为零; (ii) 加强对过去历史信息(学到的知识)的利用; (iii) 通过算法的内部状态,或根据定性的分析来判断蚁群当前的搜索能力,从而通过对信息素更新、蒸发系数的改变、释放的信息素量 Q 等的改变来避免停滞现象。下面从完全不同的角度,提出了一种基于混合行为的蚁群算法(hybrid behavior based ant colony algorithm,简称 HBACA)。

二、HBACA 算法的原理与实现

(一) 蚂蚁行为对 ACO 算法的影响

ACO 算法实际上是一种正反馈原理和某种启发式相结合的产物^[1,14,4]。算法在早期迭代过程中,因各路径上信息素量差别不大,蚂蚁主要根据城市之间的距离信息(启发式信息)来寻找较好解,这时的 ACO 算法等价于贪心算法;当算法迭代到一定代数后,较好路径上的信息素量明显高于其他边上的,此时蚁群主要通过信息素的交互及通信来寻找较好解,即信息素越强的路径成为最优路径的可能性越大。算法在此阶段的搜索过程主要利用了正反馈原理,该过程旨在强化性能较好的解,却容易导致停滞现象。从以上对蚂蚁行为的分析不难发现:基本 ACO 算法在寻优过程中,很大程度上受早期发现的较好解的影响,这些较好解以极大的概率引导蚁群走向局部最优解,这就是 ACO 算法容易出现停滞现象的主要原因。解决 ACO 算法中的停滞现象和保证其具有较强的探索能力的方法就是在“探索”和“利用”(Exploration & Exploitation)之间建立一个平衡点。也就是说既要使得算法的搜索空间尽可能的大,以寻找那些可能存在最优解的解空间;同时也要充分利用有效的历史信息(学到的知识),使算法搜索的侧重点放在那些可能具有最优解的空间,从而以更大的概率收敛到全局最优解。

(二) HBACA 算法的模型及实现

与绝大多数 ACO 算法不同,HBACA 算法中采用具有不同行为特征的蚂蚁,其模型如图 11-11 所示。

HBACA 模型由四部分组成。图中的 ①为蚂蚁,每只蚂蚁可以拥有自己的行为特征,该行为由规则集(rule set)④、储存在环境(environment)③中的知识以及

状态空间 (state space)②一起来决定。状态空间与待求解问题具有一一映射关系, 蚂蚁在其中移动, 逐步构造问题的可行解。图中蚂蚁间的虚箭头表示蚂蚁之间不能进行直接通讯, 而是在移动过程中通过环境间接通讯。蚂蚁利用通道 (channel) 读取 / 写入信息到环境, 环境储存了蚁群过去行动中获取的历史知识。下面首先给出蚂蚁行为的定义。

定义 11.5.1 蚂蚁行为: 蚂蚁在前进过程中, 用以决定其下一步移动到哪一个状态的规则集合。通常情况下, 蚂蚁行为可表示为 $\text{action} = \{\Omega_i | i = 1, 2, \dots, n\}$, 其中 Ω_i 为影响蚂蚁行为的第 i 个因素。

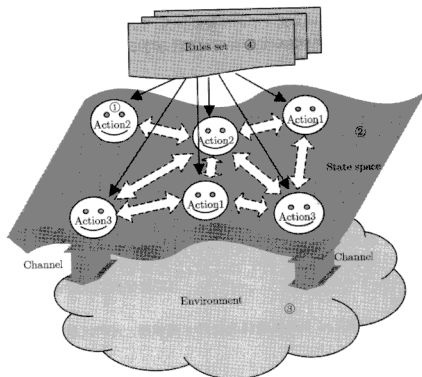


图 11-11 HBACA 算法的模型

由于 HBACA 算法中影响蚂蚁行为的因素有规则集、环境、状态空间等, 结合定义 11.5.1, 在本算法中定义如下四种类型的蚂蚁行为:

- (1) 蚂蚁行为 1(action1): 蚂蚁以随机方式选择下一个要到达的状态;
- (2) 蚂蚁行为 2(action2): 蚂蚁以贪婪方式选择下一个要到达的状态, 即

$$p_{ij}^k = \begin{cases} \frac{\eta_{ij}}{\sum_{s \in J_k(i)} \eta_{is}}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则;} \end{cases} \quad (11.5.1)$$

(3) 蚂蚁行为 3(action3): 蚂蚁按式 (11.5.2) 选择下一个要到达的状态

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}}{\sum_{s \in J_k(i)} \tau_{is}}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则;} \end{cases} \quad (11.5.2)$$

(4) 蚂蚁行为 4(action4): 蚂蚁按式 (11.5.3) 选择下一个要到达的状态

$$j = \arg \max(\tau_{is} \cdot \eta_{is}), s \in J_k(i). \quad (11.5.3)$$

由于 HBACA 算法中共定义了四种蚂蚁行为, 因此可将蚁群按蚂蚁行为分成四个子蚁群, 每个子蚁群中的蚂蚁具有相同的行为特征. 设各子蚁群中蚂蚁数的比例为 $r_1 : r_2 : r_3 : r_4$.

当所有蚂蚁完成解的构造过程后, 计算本次迭代的最优解 (IBS), 然后比较 IBS 与当前最优解 (GBS), 如果 IBS 的值小于 GBS, 则用 IBS 替换 GBS, 之后所有边上的信息素将进行更新.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + Q/L^{gb}, \quad (11.5.4)$$

其中, L^{gb} 为当前最优解, $\rho(0 < \rho < 1)$ 为信息素蒸发系数.

HBACA 算法的可描述为

Step 1 初始化 HBACA 算法的参数 $r_1 : r_2 : r_3 : r_4, \rho, Q, NC_{\max}, \tau_{ij}(0)$;

Step 2 生成 m 只蚂蚁, 其中 $\frac{m \cdot r_1}{r_1 + r_2 + r_3 + r_4}$ 只蚂蚁按规则 action1 行动, 另外 $\frac{m \cdot r_2}{r_1 + r_2 + r_3 + r_4}$ 、 $\frac{m \cdot r_3}{r_1 + r_2 + r_3 + r_4}$ 、 $\frac{m \cdot r_4}{r_1 + r_2 + r_3 + r_4}$ 只蚂蚁分别按规则 action2, action3, action4 行动;

Step 3 将所有 m 只蚂蚁随机地放到 n 座城市, 并将该城市的索引添加到该蚂蚁的禁表之中;

Step 4 for 每只蚂蚁 do

repeat

按自己的行为规则选择下一座城市;

移动到下一座城市;

将该城市的索引加入自己的禁忌表;

until 不能再向前移动

end for

Step 5 计算每只蚂蚁的路径的长度并找出 IBS, 如果 IBS 比 GBS 优, 用 IBS 替代 GBS;

Step 6 按式 (11.5.4) 更新边上的信息素;

Step 7 置 $NC := NC + 1$;

```

if  $NC > NC_{\max}$  then
    输出最优解;
    退出算法.
else
    清除所有蚂蚁禁忌表中的数据;
    转 Step 4.
end if

```

三、实验结果

实验数据全部来自 TSP 库中的 benchmark 实例 (<http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>), 采用编程语言 VC++6.0 在奔腾 III 处理器 (733 MHz) 上进行。

(一) HBACA 算法的参数研究

在大多数 ACO 算法中, 参数 Q 对算法性能的影响不大, 在实验中 Q 的值取 100。下面考虑 HBACA 算法中蒸发系数 ρ 和各子蚁群蚂蚁数的比例 $r_1:r_2:r_3:r_4$ 两个参数。首先在固定蒸发系数 ρ 的情况下, 用不同群体比例 $r_1:r_2:r_3:r_4$ 求解 eil51 问题, 实验结果如表 11-6 所示。表 11-6 中的“Best”、“Avg.”、“Worst”、“Std. Dev.”分别表示 15 次运行的最好解、平均值、最差解和标准方差。

表 11-6 参数 $r_1:r_2:r_3:r_4$ 对 HBACA 算法的影响

$r_1:r_2:r_3:r_4$	Best	Worst	Avg.	Std. Dev.
0.05:0.1:0.25:0.6	426	436	429.263	1.950
0.05:0.1:0.4:0.45	426	432	428.731	1.259
0.05:0.2:0.5:0.25	427	441	431.379	2.826
0.1:0.1:0.2:0.6	427	443	433.175	3.084
0.1:0.1:0.4:0.4	428	442	432.943	3.183
0.1:0.2:0.4:0.3	428	443	432.691	3.329
0.2:0.1:0.2:0.5	434	452	446.385	6.845
0.2:0.2:0.3:0.3	432	449	448.578	7.073
0.2:0.3:0.3:0.2	436	472	451.658	11.546

从表 11-6 中的数据可以看出, 蚁群中 r_1 (随机搜索) 所占比例不能太大, 否则对算法的搜索不利, 由于启发式因子在算法后期所起的作用减弱, 故 r_2 也不应该太大, 从表中的四个统计量来看, $r_1:r_2:r_3:r_4$ 取 0.05:0.1:0.4:0.45 时, 算

法具有较高的性能. 下面看 ρ 对 HBACA 算法的影响.

(二) 对比实验研究

下面利用 HBACA 算法的最优参数, 通过不同规模的 TSP 实例比较 HBACA 算法与 AS 算法之间的性能. 其中 AS 算法的参数设置为 $\alpha = 1.0$, $\beta = 5.0$, $\rho = 0.50$, $Q = 100$, HBACA 算法的参数 $\rho = 0.05$, $r_1 : r_2 : r_3 : r_4 = 0.05 : 0.1 : 0.4 : 0.45$. 在求解 Eil51、Eil75 问题时, 两种算法的蚂蚁数都为 35; 在求解 d198、KroA100 时, 两种算法的蚂蚁数为 80; 在求解 att532 时, 蚂蚁数为 300. 表 11-8 为运行结果, 其中对每一个 TSP 实例, 两种算法都运行 15 次后取其平均值, 每次迭代次数为 3000.

表 11-7 参数 ρ 对 HBACA 算法的影响

ρ	Best	Worst	Avg.	Std. Dev.
0.02	426	439	429.316	1.831
0.05	426	432	427.931	1.173
0.10	428	446	432.732	2.332
0.2	428	451	436.427	7.084
0.3	431	459	438.372	8.578
0.4	431	455	437.691	9.839
0.5	432	461	442.462	11.364
0.6	434	489	444.537	16.456
0.7	435	472	441.473	12.463
0.8	439	469	446.578	13.863

表 11-8 HBACA 算法与 AS 算法的比较 (* 代表 HBACA, # 代表 AS)

TSP Instance	Best		Avg.		Worst		Std. Dev.	
	*	#	*	#	*	#	*	#
Eil51	428.1	434	431.2	439.3	433	456	3.45	9.24
Eil75	535	546	539.2	556.8	545	569	4.61	12.17
d198	15974	16608	16032.9	16832.3	16156	17123	79.43	249.73
KroA100	21282	21429	21453.7	21980.4	21763	22848	213.69	692.92
Att532	28131	29064	28293.5	30071.6	28512	31927	192.36	1212.67

从表 11-8 中最优解和平均结果的情况可以看出, HBACA 算法明显优于 AS 算法, 标准方差显示 HBACA 算法更加稳定.

第六节 本章小结

本章介绍了基本 ACO 算法求解 TSP 时的实现, 并对影响其性能的诸多因素如参数 α, β, ρ, Q 、蚂蚁数 m 、蚂蚁的初始分布、蚁群的协同工作等对算法的影响

进行了讨论;分析了几种经典的改进 ACO 算法,同时对各种算法进行了比较;提出了两种改进的 ACO 算法,即自适应蚁群算法(AACA)和基于混合行为的蚁群算法(HBACA),模拟实验对这两种算法与 ACS 算法、AS 算法等进行了比较,结果显示这两种算法具有极高的性能。

参 考 文 献

- [1] Dorigo M, Maniezzo V, and Colomi A. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991
- [2] Dorigo M. Optimization, Learning and Natural Algorithms(in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, IT, 1992
- [3] Colomi A, Dorigo M, and Maniezzo V. Distributed optimization by ant colonies. In Proceedings of the First European Conference on Artificial Life. Elsevier, 1992. 134~142
- [4] Dorigo M, Maniezzo V, and Colomi A. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- [5] Dorigo M and Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- [6] Stützle T and Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem. In Baeck T, Michalewicz Z, and Yao X, editors. Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference. IEEE Press, 1997. 309~314
- [7] Stützle T and Hoos H. Improvements on the ant system: Introducing MAX-MIN ant system. In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms. Wien: Springer Verlag, 1997. 245~249
- [8] Stützle T and Hoos H. MAX-MIN Ant system and local search for combinatorial optimization problems. In S. Voß, S. Martello, Osman I H, and Roucairol C, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Boston: Kluwer, 1998. 137~154
- [9] Stützle T. MAX-MIN Ant System for Quadratic Assignment Problems. Technical Report AIDA-97-04, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germany, July 1997
- [10] Bullnheimer B, Hartl R F, and Strauss C. A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics, 1999, 7(1): 25~38
- [11] Gambardella L M and Dorigo M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Prieditis A and Russell S, editors. Proceedings of the Twelfth International Conference on Machine Learning (ML-95). Palo Alto, CA: Morgan Kaufmann Publishers, 1995. 252~260
- [12] Dorigo M and Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66
- [13] 胡小兵, 黄席樾, 张著洪. 一种新的自适应蚁群算法及其应用. 计算机仿真, 2004, 21(6): 108~112

- [14] Dorigo M, Di Caro G, and Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, 5(2): 137~172
- [15] Blum C, and Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. Technical Report TR/IRIDIA/2001-13, October 2001
- [16] HU Xiao-bing, HUANG Xi-yue. Ant colony algorithm based on hybrid behavior apply to traveling salesman problem. In Jianping L, editors. *The Proceedings of the International Computer Congress 2004 on Wavelet Analysis and Its Applications, and Active Media Technology (ICWAA04-ICAMT04)*, Singapore: World Scientific Publishers, 2004. 583~588
- [17] Gambardella L M and Dorigo M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Prieditis A and Russell S, editors. *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*. Palo Alto, CA: Morgan Kaufmann Publishers, 1995. 252~260
- [18] Dorigo M and Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53~66
- [19] 覃刚力, 杨家本. 自适应调整信息素的蚁群算法. *信息与控制*. 2002, 31(3): 198~201
- [20] 王颖, 谢剑英. 一种自适应蚁群算法及其仿真研究. *系统仿真学报*. 2002, 14(1): 32~33
- [21] 吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法. *计算机研究与发展*. 1999, 36(10): 1240~1245

第十二章 蚁群优化的并行实现

第一节 蚁群优化的并行实现概述

顺序执行算法的计算复杂性阻碍了其在大规模问题中的应用. 为减少 ACO 算法的计算时间, 对其进行并行实现是一个好的解决办法. ACO 面向群体、间接通讯的特性使其天生具有并行实现的优点. 在该算法中至少可以使用三种不同的并行策略: (i) 蚂蚁级并行策略. 该并行策略最为直观, 即用 NC ($NC > 1$) 个蚁群分别求解问题的解, 在算法执行过程中, 蚁群之间可以交换 / 不交换信息. (ii) 数据级并行策略. 该策略将待求解问题按数据域分解成许多子问题, 每个子问题由一群蚂蚁求解. (iii) 函数级并行策略. 该策略让 ACO-MH 中的三个过程 (ants_generations_and_activity、pheromone_evaporation 和 daemon_actions) 并行运行, 必要时可以交换一些同步信息. 显然, 函数级并行策略可以和另外两种并行策略组合使用. 目前, 绝大多数并行 ACO 算法都使用蚂蚁级并行策略^[1~5].

Bolondi 和 Bondanza^[1] 通过将每只蚂蚁分配到单个的处理机单元上, 在 CM-2 (Connection Machine) 上实现了第一个求解 TSP 的并行 AS 算法. 不幸的是, 该方法在细粒度并行机上的实验结果显示蚂蚁之间的通讯量过大. 由于蚂蚁将主要时间花在与其它蚂蚁的通讯 (通过修改信息素的迹) 上, AS 算法的性能并没有得到明显改善, 当问题规模越大时, 其性能越差.

另外, Bolondi 和 Bondanza^[1] 在 16 transputers 网络上的粗粒度并行实现获得了较好的结果. 在该实现中, 将蚂蚁分成 NC 个子蚁群 (NC 指可利用的处理机数目), 每个子蚁群实现成标准的 AS 算法并以完整的蚁群运行. 当每个子蚁群完成一次迭代后, 通过一个分层次的广播通讯处理过程收集所有 NC 个子蚁群的周游长度, 之后将收集到的信息广播到所有的子蚁群. 同时, 使所有子蚁群中信息素的迹同步更新. 该并行实现方式获得的加速度与处理器的个数 NC 呈线性关系, 同时当问题的规模增大时, 表现较为稳定.

最近几年, Bullnheimer, Kotsis 和 Strauss^[3] 提出了两种粗粒度的 AS 算法的并行模型, 分别称为同步并行实现 (synchronous parallel implementation, 简称 SPI) 和部分异步并行实现 (partially asynchronous parallel implementation, 简称 PAPI), SPI 基本上与 Bolondi, Bondanza 在 Transputers 上实现的相同; 在 PAPI 中, 经过一定迭代次数后, 各子蚁群之间交换信息. 模拟实验显示在时间和加速度方面 PAPI 比 SPI 具有更高的性能, 其原因可能是因为在 PAPI 中减少了子蚁群之间的通讯. 有关解的质量的比较还需要进一步的实验结果来验证.

SPI 算法首先由初始进程 (主进程) 创建一组从进程, 每个从进程对应 AS 算法中的一只蚂蚁. 主进程分发初始化信息 (距离矩阵 D 和初始信息素的迹 τ_0) 后, 每个子进程独立地构造自己的路径, 并在路径构造完成后计算其周游长度. 当所有的子进程完成这些工作后, 将其构造的路径和周游长度发送给主进程. 主进程计算本次迭代的最优解并更新信息素的迹, 然后将信息素的迹分发给每一只蚂蚁, 重新初始化信息素后开始新的迭代.

如果忽略进程 (蚂蚁) 间的通讯费用, 该并行实现隐含着最优的加速 (speedup) 性能. 假如有足够的处理器, 即每个进程可以分配一个处理器, 则有

$$S_{asymptotic}(m) = \frac{T_{seq}(m)}{T_{par}(m, \infty)} = \frac{O(m^3)}{O(m^2)} = O(m), \quad (12.2.1)$$

其中, $T_{seq}(m) = O(m^3)$ 是规模为 m 的问题顺序执行的时间复杂度^[7], $T_{par}(m, \infty) = O(m^2)$ 为规模为 m 的问题并行执行的时间复杂度, 该式假定系统具有无限的资源.

通常情况下, 系统处理机数目 N 远小于问题的规模 (因为 AS 算法中蚂蚁数和问题的规模相等时算法有较高的性能, 故 N 也远小于蚂蚁数 m). 因此, 在实际并行实现中, 多个进程 (蚂蚁) 将被分配到一个处理机上, 从而增大了系统的粒度. 为了平衡各处理机, 可让每个处理机上分配相同数量的蚂蚁, 即对蚂蚁 $m_i (i = 1, 2, \dots, m)$, 按 $(m_i : j = i \bmod N)$ 分配到处理机 $j (j = 1, 2, \dots, N)$ 上. 当考虑通讯费用时, 分配到处理机的计算量和处理机之间的数据通讯量必须达到平衡. 在 SPI 算法中, 通讯量和通讯频率非常高. 每次迭代完成后, 所有周游的路径以及它们的长度都必须发送给主进程, 主进程在更新信息素的迹之后, 再将其广播给所有的从进程, 这样才完成了一次迭代. 这部分的同步和通讯费用 $T_{ovh}(m, N)$ 通常要降低系统的性能, 于是加速度减小为

$$S(m, N) = \frac{O(m^3)}{O(m^3/N) + T_{ovh}(m, N)}, \quad (12.2.2)$$

对通讯费用 $T_{ovh}(m, N)$ 的估计取决于并行实现时的具体通讯行为.

为减少各进程间的通讯频率, 提高并行实现的性能, Bullnheimer^[3] 提出了部分异步并行实现 (PAPI). 如图 12-2 所示, 每个处理机上运行一定数量的工作进程, 这些工作进程独立地执行顺序算法的迭代过程, 经过一定代数以后, 所有的工作进程需要进行一次全局的同步, 即主进程将对信息素的迹进行全局更新. 因此, 在 PAPI 中减少了子进程之间通讯的频率, 从而降低了子进程间的通讯费用, 但每次局部迭代中好的解可能被其他的工作进程忽略代, 因此, 在选择局部 / 全局迭代率时还应慎重.

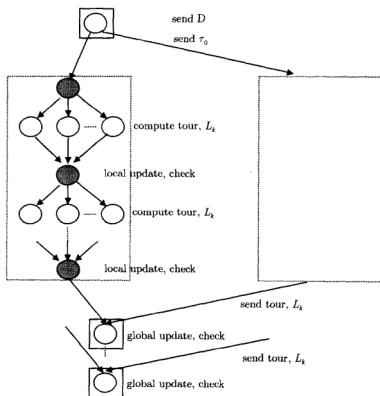


图 12-2 蚂蚁系统的部分异步并行实现 (Bullnheimer, 1997)

第三节 SPI 与 PAPI 的对比实验

一、SPI 和 PAPI 相对计算量、通讯和闲置时间

图 12-3 表示 SPI 算法中的相对工作 (busy)、通讯 (comm) 和空闲 (idle) 时间的关系。工作时间包含所有计算工作的时间；通讯时间包含比较、发送消息和数据所需要的时间；空闲时间为工作时间和通讯时间之外的时间，即等待消息、通讯的过程所占用的时间。工作、通讯和空闲时间之和构成所有执行时间。

从图 12-3 中可以看出，对于小规模的问题（规模为 50），空闲时间占居优势，这是因为频繁等待消息造成的。随着问题规模的增大（规模为 250），工作时间所占比例显著提高，而通讯时间可以忽略不计。

图 12-4 显示 PAPI 算法的行为特征与 SPI 算法非常相似，即空闲时间随着问题规模的增大而减小，随着处理机数目的增大而增大。但与 SPI 算法相比较，由于 PAPI 算法降低了各进程间的通讯频率，从而减小了空闲时间。对于小规模的问题

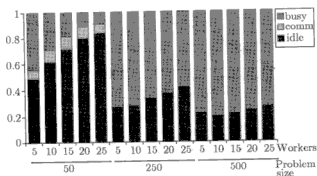


图 12-3 SPI 中的相对计算、通讯和空闲时间

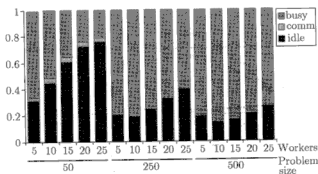


图 12-4 PAPI 中的相对计算、通讯和空闲时间

题，通讯时间甚至可以忽略不计，实验中的局部迭代次数与全局迭代次数之比为 5/1。

二、SPI 和 PAPI 中加速度、效率和功效比较

在 SPI 和 PAPI 算法中，计算 (busy) 时间随处理机数目的增多而减少说明其利用率降低。为找到最优的处理机数，需在加速度 (speedup) 增加和效率 (efficiency) 减小之间找到平衡点。图 12-5 显示了加速度、效率和功效 (efficacy) 与处理机数目之间的关系。

从图 12-5 可以看出，对于小规模的问题而言，加速度非常低，且随着工作进程的增加逐渐减小，致使算法的效能接近零；对于大规模问题而言，加速度 S 接近优化值 $S(N) \approx N$ (其中 N 为处理机数目)，效能以非常慢的速度减小。对于图中的有效性曲线，当同时考虑加速度和算法效能的情况下，有效性曲线的最大值即为工作进程的最优数。图中显示工作进程的最优数为 25。

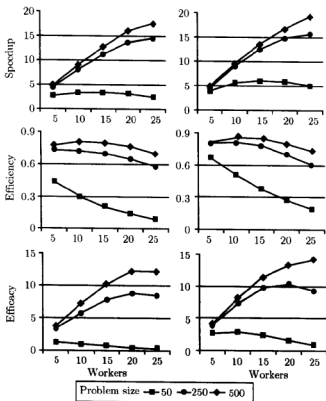


图 12-5 SPI 算法 (左) 和 PAPI 算法 (右) 的加速度、效率和功效比较

第四节 对一类带聚类特征 TSP 的 并行蚁群算法求解^[9]

一、引言

在求解小规模 TSP (或其他优化问题) 时, ACO 算法表现出极高的性能^[7,8], 但随着问题规模的增大, 其收敛速度明显减慢。图 12-6 是 ACS 算法 (目前公认的收敛速度较快的 ACO 算法) 求解 TSP eil51, pr107, pr124, pr152, pr226, pr299, pcb442, rat783, pr1002 时城市数与收敛时间之间的关系。从图中可以看出, 随着城市数目的增大, 算法找到目前已知的最优解所需的时间急剧增加, 故采用顺序 ACO 算法求解大规模 TSP 几乎不太可能。

为了解大规模 TSP, 人们提出了一些并行 ACO 算法^[1~5]。在 ACO 算法的三种并行实现方式中, 目前大多采用蚂蚁级并行实现策略。这类并行实现较为成功的例子有 PAPI 算法, 但当待求解问题的规模较大时, 受处理机数目的限制, 进程

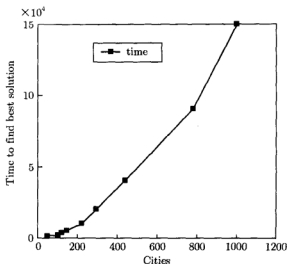


图 12-6 ACS 算法求解 TSP 时城市数与收敛时间之间的关系

间的通讯仍然是 PAPI 算法的瓶颈。

作者针对有明显聚类特征的大规模 TSP，充分利用问题本身所具有的特征，提出了一种带聚类处理的蚁群算法 (Clustering Processing Ant Colony Algorithm，简称 CPACA)。该算法在数据级对大规模 TSP 进行聚类处理，将大规模 TSP 分解成许多小规模子问题，然后对每一个小规模子问题采用 AS 算法并行求解，以此降低问题的复杂度，并通过并行求解的方式提高算法的求解速度。实验结果显示，CPACA 算法在求解此类特殊 TSP 时非常有效。

二、算法设计与实现

(一) CPACA 算法的原理

ACO 算法在求解小规模 TSP (如 30 座城市的 TSP Oliver30) 时具有极高的性能^[7]。如果能将大规模 TSP 分解成一些小规模的子问题，然后对每一个小规模子问题分别使用 ACO 算法并行求解，最后再将每个小规模子问题的解按一定的规律合并成待求解问题的解，这将大大提高算法的性能。如何分解大规模 TSP 成为解决该问题的关键。

为了找出分解大规模 TSP 的规律，首先人为地构造两类具有不同分布特征的 TSP。图 12-7 为球状分布的 TSP，可以聚成 4 类，图 12-8 为线状和球状的混合分布，可以聚成 3 类。用 ACS 算法求解这两个 TSP，当算法迭代到 1000 次，其结果如图 12-7、图 12-8 所示。

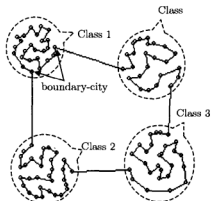


图 12-7 110 城市的 TSP, 使用 ACS 算法迭代 1000 次后的结果

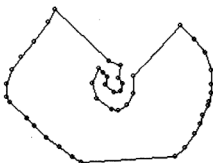


图 12-8 45 城市的 TSP, 使用 ACS 算法迭代 1000 次后的结果

通过对图 12-7、图 12-8 中 TSP 解的结构进行分析, 发现在带聚类特征 TSP 的最优解 (近优解) 中, 有如下规律:

(1) 设规模为 n 的 TSP 按城市的分布可聚成 c 类, 类 i ($i = 1, 2, \dots, c$) 中的城市数为 w_i , 显然有 $\sum_{i=1}^c w_i = n$. 则在类 i 中, 有且只有两座城市 $u_1^i, u_{w_i}^i$ 分别与另外两个不同的类相连, 此处的城市 $u_1^i, u_{w_i}^i$ 被称为类 i 的边界城市 (boundary-city). 如图 12-7 中的类 1, 只有两座边界城市与类 2 和类 4 有边相连.

(2) 在类 i ($i = 1, 2, \dots, c$) 中, 有一条从边界城市 u_1^i 经过剩下的城市一次且仅一次最后到达边界城市 $u_{w_i}^i$ 的路径, 该路径是待求解 TSP 解的组成部分, 且是类 i 中从城市 u_1^i 到城市 $u_{w_i}^i$ 的最短路径. 因为如果该路径不是最短的, 则待求解 TSP 的解也非最优.

(3) 如果将类 i ($i = 1, 2, \dots, c$) 本身看成一座超级城市 i , 则由这 c 个超级城市 (类) 构成的 TSP 的解即为所有 i 个类的连接顺序. 如图 12-7 所示, 4 座超级城市的 TSP 的解为: 城市 1 (类 1) \rightarrow 城市 2 (类 2) \rightarrow 城市 3 (类 3) \rightarrow 城市 4 (类 4) \rightarrow 城市

1(类 1). 如果按该顺序将每个类中的最短路径连接起来, 则构成了待求解问题的一个可行解.

显然, 图 12-8 中具有不同分布特征的 TSP 也满足以上的 3 条规律. 从以上的分析中受到启发, 在求解带聚类特征的大规模 TSP 时, 首先可对其进行聚类处理, 并按规律 3) 确定各类的连接顺序, 然后找出各类的边界城市, 并对类 $i(i=1, 2, \dots, c)$ 采用 AS 算法并行求解, 生成一条从边界城市 u_i^1 到边界城市 $u_{w_i}^i$ 的最短路径, 最后将所有类中的最短路径按连接顺序合并起来, 便构成了待求解 TSP 的一个可行解.

(二) CPACA 算法的实现

(1) 聚类算法的选择

在 CPACA 算法中, 首先需根据 TSP 中城市分布特征选择不同的聚类算法, 且应为基于距离的聚类算法 (distance-based clustering algorithm)^[10]. 对于类内模式为球状分布的 TSP, 选择 C -均值法, 图 12-7 即为这种情况; 对于类内模式为非球状问题, 如条状或线状分布特征的 TSP, 选用近邻函数法^[9].

(2) 确定类连接顺序

为了将各类中的最短路径最终合并成待求解问题的解, 必须先求出各类的连接顺序. 如图 12-7 所示, 在 TSP 的最优解中, 各类的连接顺序为: 类 1 → 类 2 → 类 3 → 类 4 → 类 1. 显然, 为使待求解 TSP 的解最优, 类与类之间所有连接边的长度之和应最小, 该问题等价于求解以每一类作为一个超级城市的 TSP 的解. 因此, 可以采用下面的方法确定各类之间的连接顺序.

设待求解 TSP 中的城市可聚成 U_1, U_2, \dots, U_c 类 (c 为类的个数), $K_i(i=1, 2, \dots, c)$ 为每一类的中心, 其中 K_i 有可能不是待求解 TSP 中的城市. 则将城市 K_1, K_2, \dots, K_c 看成一个 TSP, 并用 AS 算法对其求解, 设求得解为 $K_{i_1}, K_{i_2}, \dots, K_{i_c}, K_{i_1}$ (其中 i_1, i_2, \dots, i_c 为 $1, 2, \dots, c$ 的一个排列), 则 $i_1, i_2, \dots, i_c, i_1$ 即为各类的连接顺序.

(3) 确定类的边界城市

设类 p 、类 $q(p, q=1, 2, \dots, c$ 且 $p \neq q)$ 按连接顺序相邻, $u_i^p(i=1, 2, \dots, w_p)$ 为类 p 中的城市, $u_j^q(j=1, 2, \dots, w_q)$ 为类 q 中的城市, 则类 p 、类 q 中的边界城市为

$$\{u_k^p, u_l^q\} = \arg \min_{\substack{i=1, 2, \dots, w_p \\ j=1, 2, \dots, w_q}} d(u_i^p, u_j^q), \quad (12.4.1)$$

其中 u_k^p 为类 p 中的一个边界城市, u_l^q 为类 q 中的一个边界城市; U_p, U_q 分别为类 p 、类 q 中的城市集合; w_p, w_q 分别为类 p 、类 q 中的城市数目.

当各类的连接顺序确定后, 对任意类 i 只有一个直接前驱类和一个直接后继类, 利用式 (12.4.1) 与其直接前驱类运算, 可以求出边界城市 u_i^1 , 通过其直接后继类, 可以求出第二个边界城市 $u_{w_i}^i$.

(4) 求解类内最短路径的 AS 算法

为求解类内从第一个边界城市经所有剩下的城市一次且仅一次最后到达第二个边界城市的最短路径, 采用改进的 AS 算法来求解. 与文献 [7] 相比, 其差别主要在最短路径的生成上. 本文中所有蚂蚁都从第一个边界城市出发, 逐步选择除第二个边界城市之外的城市, 生成问题的可行解. 状态转移规则、信息素更新规则都与 ant-cycle 模型相同.

(5) 局部搜索策略

初步的实验结果显示, CPACA 算法在搜索解的过程中, 能以极快的速度进入最优解所在的子空间, 但其后的搜索速度相对较慢. 为此, 在该算法中引入了局部搜索策略 2-opt, 以便更好地发现局部空间中的较好解. CPACA 算法描述如下:

Step 1 根据 TSP 中城市分布的先验知识, 选用 C -均值或近邻函数法对 TSP 中的城市进行聚类处理, 设可将城市聚成 U_1, U_2, \dots, U_c 类 (其中 c 为类的个数), 且 $K_i (i = 1, \dots, c)$ 为每一类的中心, 其中 K_i 可以不是 TSP 中的城市;

Step 2 将 $K_i (i = 1, \dots, c)$ 看成一个 TSP, 使用 AS 算法对其求解, 设求得的解为 $K_{i_1}, K_{i_2}, \dots, K_{i_c}, K_{i_1}$, 此解即为各类中最短路径的连接顺序;

Step 3 根据 (12.4.1) 式计算各类中的边界城市;

Step 4 对每一类 $U_i (i = 1, 2, \dots, c)$ 中的城市, 采用改进的 AS 算法并行求解从边界城市 $u_{i_1}^i$ 到边界城市 $u_{w_i}^i$ 的一条经过所有剩下城市一次且仅一次的最短路径 $R_i (i = 1, 2, \dots, c)$;

Step 5 对所有的类, 按连接顺序 $i_1, i_2, \dots, i_c, i_1$ 通过类与类之间的边界城市连接起来, 便构成了待求解 TSP 的一个可行解;

Step 6 对所求得解进行 2-opt 局部领域搜索, 并记录本次迭代的最优解;

Step 7 将所得解与当前最优解进行比较, 如优于当前最优解, 则用其替换当前最优解;

Step 8 检查结束条件是否满足, 如果满足, 则退出算法; 否则转步骤 2.

实验结果显示, CPACA 算法对带聚类特征的 TSP 具有极高的性能, 其时间复杂度分析如下: 设待求解 TSP 由 n 个城市组成, 且可聚成 c 类, 假设每类中城市数相等, 则每类中的城市数为 n/c . 由文献 [7] 知, 顺序 AS 算法的时间复杂度为 $O(NC \cdot n^3)$, 其中 NC 为算法迭代的次数, 则 CPACA 算法由 c 个规模为 n/c 的子问题并行执行, 故时间复杂度为 $O(NC \cdot (n/c)^3 \cdot c)$.

由

$$\frac{AS(n_{AS}, NC_{AS})}{CPACA(n_{CPACA}, NC_{CPACA})} = \frac{O(NC_{AS} \cdot n^3)}{O(NC_{CPACA} \cdot (n/c)^3 \cdot c)} = O\left(\frac{NC_{AS}}{NC_{CPACA}} \cdot c^2\right)$$

可以看出, AS 算法的时间复杂度是 CPACA 算法的 $O\left(\frac{NC_{AS}}{NC_{CPACA}} \cdot c^2\right)$ 倍, 其中 NC_{AS} 是 AS 算法的迭代次数, NC_{CPACA} 是 CPACA 算法中求解子问题的 AS 算

法的迭代次数。 NC_{AS} 一般情况下远大于 NC_{CPACA} , 所以 CPACA 算法的效率至少提高了 c^2 倍。显然, 当问题可分的类越多, CPACA 算法的效率越高, 但如果问题的聚类特征不明显, 则 CPACA 算法退化为顺序 AS 算法。

在 CPACA 算法中并行运行的 AS 算法之间不需要任何通讯, 故不存在蚂蚁级并行策略中的通讯瓶颈问题。

三、实验结果

实验选用 TSP 库中的实例 pr107、pr136、d2103、u2319、pr2392, 通过 ACS 算法与本文提出的 CPACA 算法进行对比实验。针对不同的 TSP 实例, ACS 算法与 CACA 算法的参数设置如表 12-1、表 12-2。

表 12-1 CPACA 算法针对不同问题实例的参数设置

问题	参数				CPACA 算法中		
	α	β	ρ	Q	类数	蚂蚁数	蚂蚁算法的迭代次数
Pr107	1.0	5.0	0.5	10.0	2	30	300
Pr136	1.0	5.0	0.5	10.0	8	10	50
d2103	1.0	5.0	0.5	10.0	36	30	300
u2319	1.0	5.0	0.5	10.0	54	20	200
pr2392	1.0	5.0	0.5	10.0	16	50	500

表 12-2 ACS 算法针对不同问题实例的参数设置

问题	参数			
	β	ρ	γ	τ_0
pr107	2.0	0.1	0.1	$(107*44303)^{-1}$
pr136	2.0	0.1	0.1	$(136*96772)^{-1}$
d2103	2.0	0.1	0.1	$(2103*8450)^{-1}$
u2319	2.0	0.1	0.1	$(2319*234256)^{-1}$
pr2392	2.0	0.1	0.1	$(2392*378032)^{-1}$

(其中 ρ 为全局更新时信息素的蒸发系数, γ 为局部更新时信息素的蒸发系数)

实验环境在内存为 128 兆, 奔腾 III 处理器 (733 MHz) 的 PC 机上的进行, 采用 Visual C++6.0 编程语言。对 pr107, pr136 问题两种算法各运行 15 次; 对 d2103、u2319、pr2392 问题两种算法各运行 4 次。表 12-3 为对比实验结果, 其中的运行时间为 ACS 算法迭代 1000 次所用的时间。

表 12-3 CPACA、ACS 算法求解不同 TSP 实例的对比结果

TSP	运行时间 (秒)	ACS 算法		CPACA 算法			已知最优解
		所求的解	误差百分比 (%)	所求的解	达到 ACS 的 解所需时间	误差百分比 (%)	
pr107	210	44661	0.81	44514	26	0.48	44303
pr136	30	100213	3.56	97218	11	0.46	96772
d2103	26000	88306	9.77	81273	1025	1.02	80450
u2319	86000	262114	11.89	245341	50	4.73	234256
pr2392	365000	426218	12.75	392698	15000	3.88	378032

分析表 12-3 中的结果可以发现, 对于同一个问题实例, CPACA 算法较 ACS 算法可以在更短的时间内找到相同质量的解, 而且这种优势随着待求解问题的规模增大和聚类数的增多而更为明显; 如果让这两个算法运行相同的时间, 则 CPACA 算法总能找到更好的解。

图 12-9 是两种算法求解 pr136 问题时的收敛曲线的比较。图中显示, 在算法运行的初期, CPACA 算法就能找到较好的解, 且在同一时刻, CPACA 算法找到解的质量总优于 ACS 算法。

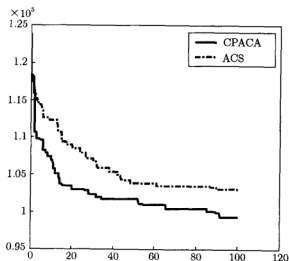


图 12-9 CPACA 算法与 ACS 算法收敛性比较

图 12-10 为 CPACA 算法找到的 pr136 问题的一个近优解。

图 12-11 是 CPACA 算法在求解 d2103 问题时运行 4 分钟后求得的解, 而 ACS 算法需要运行 3.5 小时才能达到与该值相近的解。

以上的实验结果显示, CPACA 算法在求解带聚类特征的 TSP 时非常有效,

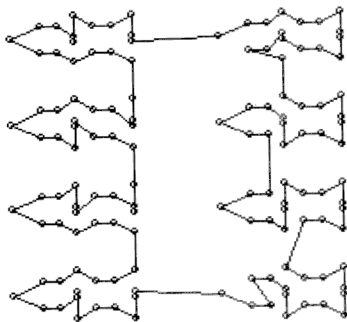


图 12-10 CPACA 算法求得 pr136 问题的解

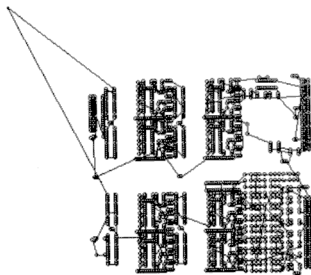


图 12-11 CPACA 算法求解 d2103 问题运行 4 分钟后的结果

当待求解问题的聚类数越多,且每个类中的城市数目较接近时,算法的性能越好;但如果问题的聚类特征不明显,则该算法将蜕化成一般的 AS 算法。由于 CPACA 算法依赖于问题的聚类特征,其实用性具有一定的限制。

第五节 本章小结

本章首先分析了 ACO 并行实现的主要途径及目前有代表性的并行 ACO 算法。然后重点介绍了 SPI 和 PAPI 并行实现算法, 并对其加速度、效率和功效进行了比较, 结果显示 PAPI 算法具有更高的性能。最后给出了作者提出一种特殊的 ACO 并行实现——CPACA 算法。该算法根据问题所具有的聚类特征对大规模 TSP 进行分解, 降低问题的规模, 从而减小问题求解时的复杂性。实验结果表明在运行的初期, CPACA 算法就能以极高的速度收敛于问题最优解的某个子空间, 从而引导算法朝正确的方向(最优解所在的子空间)进行搜索。CPACA 算法对问题的某些先验知识如聚类数目、城市的分布、聚类算法的选取等需要人来确定, 为增加算法的灵活性, 可以选用更好的聚类算法。在 CPACA 算法中, 当问题的聚类数越多, 且每类中的城市数较接近时, 算法的性能越好; 但当问题的聚类特征不明显, 该算法将退化成一般的 ACO 算法, 故算法的实用性有一定限制。

参 考 文 献

- [1] Bolondi M and Bondanza M. Parallellizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore. Master's thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1993
- [2] Dorigo M. Parallel ant system: An experimental study. Unpublished manuscript, 1993
- [3] Bullnheimer B, Kotsis G, and Strauss C. Parallelization strategies for the ant system. Technical Report POM 9-97, Institute of Management Science, University of Vienna, Austria, 1997
- [4] Krüger F, Merkle D, and Middendorf M. Studies on a parallel ant system for the BSP model. Unpublished manuscript
- [5] Stützle T. Parallelization Strategies for Ant Colony Optimization. Research report AIDA-98-03. Darmstadt: Department of Computer Science, Darmstadt University of Technology, 1998
- [6] Middendorf M, Reischle F and Schmech H. Information Exchange in Multi Colony Ant Algorithm. In Proceedings of IEEE Symposium on Parallel and Distributed Processing. Third Workshop on Biologically Inspired Solution to Parallel Processing Problems, IPDPS 2000, 2000. 645~652
- [7] Dorigo M, Maniezzo V, and Colomi A. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- [8] Dorigo M, Di Caro G, and Gambardella L M. Ant algorithms for discrete optimization. Artificial Life, 1999, 5(2): 137~172
- [9] 胡小兵, 黄席樾. 对一类带聚类特征 TSP 问题的蚁群算法求解. 系统仿真学报, 2004, 16(12): 2683~2686
- [10] 孙仰祥. 现代模式识别. 长沙: 国防科技大学出版社, 2002. 460

第十三章 蚁群优化算法的应用

第一节 概 述

自从 M Dorigo^[1] 等人首先将蚂蚁算法应用于旅行商问题 (TSP) 并获得了较大的成功后, 该算法吸引了许多学者对其进行研究, 并取得了一系列的研究成果。除了 TSP, 蚂蚁算法还用于求解二次分配问题 (quadratic assignment problem, 简称 QAP)^[2,3]、车间作业调度问题 (job-shop scheduling problem, 简称 JSP)^[4]、背包问题 (knapsack problem, 简称 KP)^[5]、车辆路径问题 (vehicle routing problem, 简称 VRP)^[6~8]、图着色 (graph coloring, 简称 GC) 问题^[9]、网络路由 (network route, 简称 NR) 问题^[10~21] 和机器人路径规划问题^[22] 等。

蚁群算法可用于解决各种组合优化问题、函数优化问题、机器人路径规划等问题, 其关键是: (1) 将待解决的问题表示成相应的赋权图; (2) 能定义一种正反馈过程 (如 TSP 中的残留信息素); (3) 问题结构本身能提供解题用的启发式信息 (如 TSP 中城市间的距离); (4) 建立约束机制 (如 TSP 中已访问城市的列表), 就可以应用蚁群算法来求解该问题。

第二节 蚁群优化算法与 K-TSP

一、K-TSP

K-TSP(K-person traveling salesman problem) 即 k 个人从城市 1 出发分头去访问 $n-1$ 个城市, 每个城市有且仅有一人到达, 最后都回到城市 1, 问怎样安排使得 k 个人的总访问路线最短?

该问题的数学形式表达为: 设 $V = (v_1, v_2, \dots, v_n)$ 平面上的 n 个点的集合, $G = (V, E)$ 是 V 上的完全图, $C: E \rightarrow R$ 为权函数。称 H 为图 $G = (V, E)$ 的 k -周游路, 如果它是 k 条子周游路的集合 $H = (H_1, H_2, \dots, H_k)$, 这里

- (1) H_i 为至少包含 3 条边的简单图, $i = 1, 2, \dots, k$;
- (2) H_i 经过顶点 v_1 , $i = 1, 2, \dots, k$;
- (3) 任给 $v \in V/\{v_1\}$, 存在唯一的子周游路 H_i 经过 v 。

k -周游路 H 的长度记为 $C(H)$, 即

$$C(H) = \sum_{i=1}^k C(H_i) = \sum_{i=1}^k \sum_{e \in H_i} C(e), \quad (13.2.1)$$

则 K-TSP 为寻找最短 k -周游路 H .

显然, 当 $k=1$ 时, 该问题就是 TSP, 因此 K-TSP 可看成更一般形式的 TSP. 目前只有少量的文献^[23,24]对 K-TSP 进行了讨论. 下面利用蚁群算法对 K-TSP 进行求解, 实验结果表明本算法是行之有效的.

二、求解 K-TSP 的蚁群算法 (ACA-KTSP)

(一) ACA-KTSP 中解的构造

在求解 TSP 的蚁群算法中, 每只蚂蚁可构造 TSP 的一个解. 鉴于 K-TSP 与 TSP 的差异, 我们采用 k 只蚂蚁来共同构造问题的一个解, 即一组蚂蚁 (共 k 只) 与 K-TSP 的一个可行解相对应. 在 ACA-KTSP 算法中, 将会有 m 组 (共 $m \times k$ 只) 蚂蚁共同协作来发现问题的最优解.

对于第 i ($i=1, 2, \dots, m$) 组的蚂蚁 j ($j=1, 2, \dots, k$), 从城市 1 出发访问 m_j^i 个城市后回到城市 1, 其中 m_j^i 是蚂蚁 j 出发之前按一定规则随机生成的整数, 表示其所能访问的城市数. 由 K-TSP 的数学定义知 m_j^i 应满足

$$\begin{cases} m_j^i \geq 2 \\ \sum_{j=1}^k m_j^i = n-1 \end{cases} \quad (i=1, 2, \dots, m). \quad (13.2.2)$$

同时, 为第 i ($i=1, 2, \dots, m$) 组的每一只蚂蚁 j ($j=1, 2, \dots, k$) 分配一个子周游列表 H_j^i ($j=1, 2, \dots, k; i=1, 2, \dots, m$), 该表记录了蚂蚁 j 当前已访问过的城市. 当 H_j^i 中的元素个数等于 m_j^i 时, 蚂蚁 j 停止搜索并后到城市 1. 当第 i 组的所有蚂蚁都回到城市 1 之后, 所有子周游的路径 $\{H_1^i, H_2^i, \dots, H_k^i\}$, ($i=1, 2, \dots, m$) 便构成了 K-TSP 的一个解. 每次迭代中 m 组蚂蚁可以生成 m 组解.

(二) 蚂蚁路径的选择

每只蚂蚁根据边上的信息素独立地选择下一座城市, 即在时刻 t , 对于第 u ($u=1, 2, \dots, m$) 组的蚂蚁 v ($v=1, 2, \dots, k$) 从城市 i 转移到城市 j 的转移概率 $p_{ij}^{uv}(t)$ 为

$$p_{ij}^{uv}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{s \in J_u} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}(t)]^\beta}, & \text{如果 } j \in J_u, \\ 0, & \text{否则,} \end{cases} \quad (13.2.3)$$

其中,

$$J_u = \{1, 2, \dots, n\} - \bigcup_{v=1}^k H_v^u$$

表示第 u 组的蚂蚁下一步允许选择的的城市, H_v^u 记录了第 u 组的第 v 只蚂蚁所走过的城市. 当 H_v^u 中的城市数等于 $m_v^u (u = 1, 2, \dots, m; v = 1, 2, \dots, k)$ 时, 蚂蚁 v 回到城市 1; η_{ij} 是一个启发式因子, 表示蚂蚁从城市 i 转移到城市 j 的期望程度, 在 ACA-KTSP 算法中 η_{ij} 取城市 i, j 之间距离的倒数; α 和 β 分别表示路径上信息素量和启发式因子的相对重要程度.

(三) 信息素的更新

当所有组的蚂蚁完成一次周游后, 最优组蚂蚁所经过路径上的信息素量根据式 (13.2.4) 调整

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}, \rho \in (0, 1), \quad (13.2.4)$$

其中 ρ 表示信息素量的蒸发系数, $\Delta\tau_{ij}$ 表示本次周游中边 ij 上信息素的增量. 如果最优组蚂蚁没有经过边 ij , 则 $\Delta\tau_{ij}$ 为零; 否则取与解的质量相关的数, 即

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{L^*}, & \text{若最优组蚂蚁经过边 } ij, \\ 0, & \text{否则,} \end{cases} \quad (13.2.5)$$

其中, Q 为常数, L^* 表示最优组蚂蚁在本次周游中走过路径的长度.

对 ACA-KTSP 算法的描述如下:

Step 1 $NC := 0$ (NC 为迭代次数), 初始化各边上的信息素 $\tau_{ij} (i, j = 1, 2, \dots, n)$, 将 $m \times k$ 只蚂蚁分成 m 组 (每组 k 只) 置于城市 1;

Step 2 对第 $i (i = 1, 2, \dots, m)$ 组的蚂蚁, 按公式 (13.2.2) 生成一组随机数 $m_j^i (j = 1, 2, \dots, k)$, 其中 m_j^i 为第 i 组的蚂蚁 j 所能经过的城市数;

Step 3 对第 $i (i = 1, 2, \dots, m)$ 组的蚂蚁 $j (j = 1, 2, \dots, k)$, 如果其子周游列表 H_j^i 中的城市数小于 m_j^i , 则按公式 (13.2.3) 生成选择概率, 并根据该值选择下一座城市; 否则蚂蚁 j 回到城市 1;

Step 4 如果步骤 3 中所有蚂蚁都回到城市 1, 转步骤 5; 否则转步骤 3;

Step 5 计算各组蚂蚁的目标函数值 $F_k (k = 1, 2, \dots, m)$, 记录当前的最好解;

Step 6 按公式 (13.2.4)、(13.2.5) 更新边上的信息素;

Step 7 $NC := NC + 1$, 若 NC 小于预定的迭代次数且无退化行为 (即找到的都是相同解), 则转步骤 2; 否则输出最优解, 算法结束.

三、实验结果

选用 51 城市的 TSPeil51 (在 <http://www.iwr.uniheidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>) 分别对不同人数 (k 值) 的 TSP 进行对比实验, 参数设置如表 13-1.

表 13-1 ACA-KTSP 算法的参数设置
(算法中的蚂蚁数为 40, 迭代的次数为 1000)

α	β	ρ	Q
1.0	5.0	0.5	10.0

表 13-2 k 取不同值时, 每种情况分别运行 10 次后的结果

k 的取值	最优解	最差解	平均值
2	449	492	459
3	468	512	475
5	527	586	539

图 13-1 为当 k 取 2, 3, 5 时解的进化情况。

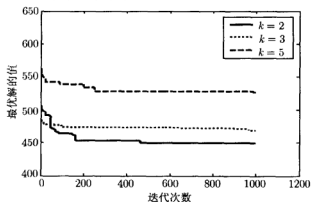


图 13-1 k 取 2, 3, 5 时, ACA-KTSP 算法解的进化曲线

图 13-2、图 13-3、图 13-4 分别为 $k = 2, 3, 5$ 时问题的解。

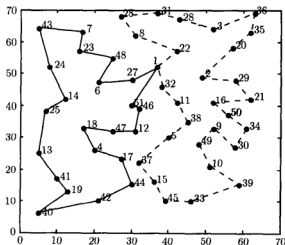
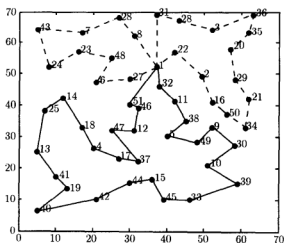
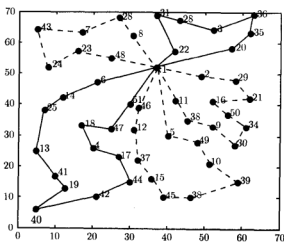


图 13-2 $k = 2$ 时问题的解

图 13-3 $k=3$ 时问题的解图 13-4 $k=5$ 时问题的解

K-TSP 算法采用 k 只蚂蚁来构造 K-TSP 的一个可行解, 再通过 m 组 (每组 k 只) 蚂蚁共同在问题的解空间搜索最优解。实践结果表明蚁群算法是一种求解组合优化问题的有效算法。

第三节 蚁群优化与二次分配问题

一、二次分配问题

二次分配问题 (quadratic assignment problem, QAP)^[2,3] 是继 TSP 之后蚁群算

法第二个成功求解的 NP- 难题。该问题具有重要的理论与实用价值,许多实际的问题如大学校园的规划、医院的布局、最短电路布线、键盘布局问题等都可以转化为 QAP 来解决。

QAP 可描述为:已知有 n 个位置和 n 家工厂,各位置之间的距离矩阵设为 $D = (d_{ij})_{n \times n}$,各工厂之间的运输量矩阵为 $F = (f_{ij})_{n \times n}$ 。现要将这 n 家工厂建造在这 n 个位置上,使得总费用最小。其中, d_{ij} 表示位置 i 与位置 j 之间的距离, f_{ij} 位置 i 与位置 j 之间的费用。故工厂 i 建造在位置点 k 且工厂 j 建造在位置点 l 所导致的费用(物料等,与具体的问题相关)为 $f_{ij} \times d_{kl}$ 。则 QAP 即:

$$z = \min \sum_{i=1}^n \sum_{j=1}^n f_{ij} \cdot d_{\pi(i)\pi(j)}, \quad (13.3.1)$$

其中, $\pi(i)$ 表示工厂 i 建造的位置,且 $\pi(i) \in \{1, 2, \dots, n\}$ 。则 $f_{ij} \cdot d_{\pi(i)\pi(j)}$ 表示工厂 i 建在位置 $\pi(i)$, 工厂 j 建在 $\pi(j)$ 时所需的费用。

各工厂的建造费用由于对问题求解的难度没有本质上的影响,故而常常忽略不计。QAP 由于目标函数的非线性而变得异常困难,可以将该问题表示成二次目标函数的 0-1 整数规划问题(这也正是二次分配问题的由来),即:

$$\sum_{i=1}^n \sum_{j=1}^n \sum_{l=1}^n \sum_{k=1}^n f_{ij} \cdot d_{kl} \cdot x_{ik} \cdot x_{jl}, \quad (13.3.2)$$

其中

$$x_{ij} \in \{0, 1\}, \quad \{i, j = 1, 2, \dots, n\}, \quad (13.3.3)$$

$$\sum_{i=1}^n x_{ij} = 1, \quad \{j = 1, 2, \dots, n\}, \quad (13.3.4)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad \{i = 1, 2, \dots, n\}, \quad (13.3.5)$$

当工厂 i 被建在位置 k 时, x_{ik} 的值为 1, 否则为 0。

为了用蚁群优化算法求解 QAP, 首先需将 QAP 表示成图 $G = (C, L)$ 的形式。其中 C 是 QAP 中工厂和位置的集合, L 为工厂与位置之间的连接构成的集合。则将工厂分配到各位置的过程可以看成是一群蚂蚁在信息素迹和局部启发式信息的指引下沿着图 G 移动, 同时蚂蚁的移动将会受到一些条件的约束, 从而使蚂蚁在移动过程中找到合法的解。为此, 可以将 QAP 表示成图 13-5 所示的情形。

在具体分配时, 可采取固定的分配方向。如将 n 个工厂分配到 n 个位置, 或者反之。确定分配顺序后, 蚂蚁重复地使用两个步骤将工厂分配到剩下的位置。即蚂蚁首先按照某种规则选择一个将要被分配的工厂, 然后将选择的工厂分配到某一个位置。该过程重复进行直到蚂蚁构造出问题的解。信息素的迹和启发式因子在这两个步骤中都将发挥作用。在选择工厂时, 信息素的迹和启发式因子用来构造分

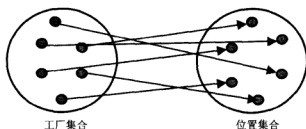


图 13-5 QAP 的图形表示

配的顺序, 该顺序将会影响算法的性能; 在分配工厂时, 信息素的迹 τ_{ij} 和启发式因子 η_{ij} 决定了将工厂 i 分配到位置 j 的期望程度。当所有的蚂蚁都生成了自己的可行解, 还可以利用局部搜索策略来提高算法的寻优能力。

二、二次分配问题与蚂蚁系统

蚂蚁系统是第一个应用于二次分配问题的蚁群优化算法, 此处记为 AS-QAP。在 AS-QAP 中, 工厂 i 按一定的概率分配到位置 j , 具有较高 τ_{ij} 和 η_{ij} 的位置将会有更大的可能性。

(一) 启发式信息

首先定义两个向量 d 和 f (潜在的好的分配定义):

$$d = [d_1, d_2, \dots, d_n]^T, \text{ 其中 } d_i = \sum_{j=1}^n d_{ij}, \quad d_{ij} \in D. \quad (13.3.6)$$

在向量 d 中, 如果 d_i 越小, 则位置 i 越靠近中心。

$$f = [f_1, f_2, \dots, f_n]^T, \text{ 其中 } f_i = \sum_{j=1}^n f_{ij}, \quad f_{ij} \in F. \quad (13.3.7)$$

在向量 f 中, 如果 f_i 越大, 则工厂 i 越重要。

通过向量 d 与 f , 定义矩阵 S 为

$$S = d \times f^T, \quad (13.3.8)$$

即 $s_{ij} = d_i \times f_j$, 其中 $S = (s_{ij})_{n \times n}$ 。

例 13.3.1 计算 S 矩阵。设 $D = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{bmatrix}$, $F = \begin{bmatrix} 0 & 60 & 50 & 10 \\ 60 & 0 & 30 & 20 \\ 50 & 30 & 0 & 50 \\ 10 & 20 & 50 & 0 \end{bmatrix}$ 。

解 因为 $d = \begin{bmatrix} 6 \\ 10 \\ 12 \\ 14 \end{bmatrix}$, $f = \begin{bmatrix} 120 \\ 110 \\ 130 \\ 80 \end{bmatrix}$, 故

$$S = d \times f^T = \begin{bmatrix} 720 & 660 & 780 & 480 \\ 1200 & 1100 & 1300 & 800 \\ 1440 & 1320 & 1560 & 960 \\ 1680 & 1540 & 1820 & 1120 \end{bmatrix}.$$

在 AS-QAP 中定义 $\eta_{ij} = 1/s_{ij}$, η_{ij} 即为将工厂 i 分配给位置 j 的启发式因子.

(二) 解的构造

在 AS-QAP 中, 首先将向量 f 中的元素按递减的排序. 在构造解的每一步, 蚂蚁 k 按概率 $p_{ij}^k(t)$ 将下一个没有分配的工厂 i 分配到空位置 j .

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{s \in N_i^k} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}]^\beta}, & \text{如果 } j \in N_i^k, \\ 0, & \text{否则,} \end{cases} \quad (13.3.9)$$

其中, $\tau_{ij}(t)$ 为 t 时刻的信息素的迹, α, β 为信息素和启发式因子的相对重要程度, N_i^k 为节点 i 的可行邻域, 即当前还没有使用的位置的集合, 显然有 $\sum_{j \in N_i^k} p_{ij}(t) = 1$.

该过程重复进行, 直到构造出问题的解.

(三) 信息素更新

当所有蚂蚁完成解的构造以后, 蚂蚁所经过路径上的信息素要进行更新, 其公式为

$$\tau_{ij}(t+n) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^m \Delta \tau_{ij}^k, \quad \rho \in (0, 1), \quad (13.3.10)$$

其中 $0 < \rho < 1$, 表示信息素的蒸发系数, ρ 可以避免信息素的无限制积累, 同时可以使算法忘掉过去部分坏的解. $\Delta \tau_{ij}^k$ 定义为

$$\Delta \tau_{ij}^k = \begin{cases} Q/J^k, & \text{如果蚂蚁 } k \text{ 将工厂 } i \text{ 分配给位置 } j, \\ 0, & \text{否则,} \end{cases} \quad (13.3.11)$$

其中 J^k 是蚂蚁 k 生成的解, Q 为常数.

三、二次分配问题与最大-最小蚂蚁系统

最大-最小蚂蚁系统 (MMAS) 是一种改进的蚂蚁系统, 该算法首先应用于 TSP, 其后又被应用于 QAP. 此处将求解 QAP 的 MMAS 算法记为 MMAS-QAP. MMAS-QAP 和 AS-QAP 的不同之处在于: (1) 在每次迭代中, 只有生成全局最优解或本次迭代最优解的蚂蚁允许更新信息素的迹; (2) 为避免搜索中出现停滞现象, 将信息素的迹的范围限定在 $[\tau_{\min}, \tau_{\max}]$; (3) 初始时刻, 所有信息素初始化为 τ_{\max} , 从而加快蚂蚁探索较好解的能力.

(一) 解的构造

在每一步构造解的过程中, 蚂蚁 k 首先选择一个没有被分配的工厂 i , 然后按照概率 $p_{ij}^k(t)$ 将工厂 i 放置到位置 j .

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)}{\sum_{l \in N_i^k} \tau_{il}(t)}, \quad \text{如果 } j \in N_i^k. \quad (13.3.12)$$

从公式 (13.3.12) 可以看出, MMAS-QAP 没有使用启发式信息. 当采用局部搜索优化解以后, 启发式信息并不能提高解的性能; 同时, 不使用启发式信息有利于调节参数.

(二) 信息素的更新

当所有蚂蚁构造完问题的解以后, 按公式 (13.3.13) 更新路径上的信息素.

$$\tau_{ij}(t+n) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}^{best}, \quad \rho \in (0, 1), \quad (13.3.13)$$

此处, $\Delta\tau_{ij}^{best}$ 定义为

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/J^{best}, & \text{如果最优解中工厂 } i \text{ 分配到位置 } j, \\ 0, & \end{cases} \quad (13.3.14)$$

其中 J^{best} 为当前算法所发现的最优解或本次迭代中的最优解. 在 MMAS-QAP 中, 如果发现搜索解的过程较慢, 则可将所有的信息素重新初始化为 τ_{\max} .

四、二次分配问题与快速蚂蚁系统 (FANT)

快速蚂蚁系统 (fast ant system, FANT) 是另一个应用于 QAP 的蚁群优化算法. 在 FANT 中, 解的构造方式与 MMAS-QAP 相同, 即也是按公式 (13.3.12) 对工厂进行分配, 且没有使用启发式信息. FANT 与其他蚁群优化算法相比, 其差别表现在: (1) 使用的蚂蚁数量; (2) 对信息素的管理.

(一) FANT 中的蚂蚁数量

FANT 是一种非群体的算法, 即在 FANT 中只使用一只蚂蚁. 单只蚂蚁可以提高算法的计算速度.

(二) FANT 中信息素的更新策略

FANT 中的信息素不会蒸发. 算法的初始时刻, 信息素被初始化为 1, 在每次迭代完成后, 按公式 (13.3.15) 更新信息素的迹.

$$\tau_{ij}(t+n) = \tau_{ij}(t) + r \cdot \Delta\tau_{ij} + r^* \cdot \Delta\tau_{ij}^{gb}. \quad (13.3.15)$$

此处 $\Delta\tau_{ij} = 1/J$, 其中 J 为本次迭代生成的目标函数值. $\Delta\tau_{ij}^{gb} = 1/J^{gb}$, 其中 J^{gb} 为当前发现的全局最优解目标函数的值. r 和 r^* 为两个参数. 初始时刻 r 的值为 1, 随着算法的运行, r 的值可能发生改变. r^* 的值保持不变. 参数 r 和 r^* 决定本次迭代的解与全局最优解的相对重要程度. 信息素的更新的方式为: (1) 如果全局最优解得到改善, 则将参数 r 设置为 1, 且所有信息素的迹都重新设置为 1, 该策略可以更加充分地利用最优解; (2) 如果全局最优解没有得到改善, 则将 r 的值减 1, 所有信息素的迹重新设置为 r , 该策略通过减小全局最优解的权重以提高算法解的多样性.

五、二次分配问题中的局部搜索

局部搜索即从某些初始的分配开始, 试图通过局部的改变来提高解的质量, 如果解的邻域中有比当前解更优的解, 则用其替代当前解.

在 QAP 中, 分配 Π 的邻域为交换 Π 中的两个工厂而得到的新的分配. 设交换工厂 $\pi(r)$ 与 $\pi(s)$ 的位置后得到的解与原来解的差值为 $\Delta(\Pi, r, s)$, 其时间复杂度为 $O(n)$, 即

$$\begin{aligned} \Delta(\Pi, r, s) = & f_{rr} \cdot (d_{\pi(s)\pi(s)} - d_{\pi(r)\pi(r)}) + f_{rs} \cdot (d_{\pi(s)\pi(r)} - d_{\pi(r)\pi(s)}) \\ & + f_{sr} \cdot (d_{\pi(r)\pi(s)} - d_{\pi(s)\pi(r)}) + f_{ss} \cdot (d_{\pi(r)\pi(r)} - d_{\pi(s)\pi(s)}) \\ & + \sum_{j=1, k \neq r, s}^n (f_{kr} \cdot (d_{\pi(k)\pi(s)} - d_{\pi(k)\pi(r)}) + f_{ks} \cdot (d_{\pi(k)\pi(r)} - d_{\pi(k)\pi(s)}) \\ & + f_{rk} \cdot (d_{\pi(s)\pi(k)} - d_{\pi(r)\pi(k)}) + f_{sk} \cdot (d_{\pi(r)\pi(k)} - d_{\pi(s)\pi(k)}). \end{aligned} \quad (13.3.16)$$

使用前次交换的差值信息, 可以更快地计算出下一次交换的差值. 设 Π' 为 Π 中交换 r 与 s 后的分配, 当再次交换工厂 u 与 v 时 (其中 $\{u, v\} \cap \{r, s\} = \emptyset$), 则交换可在常数时间内完成, 即

$$\begin{aligned} \Delta(\Pi', r, s) = & \Delta(\Pi, r, s) + (f_{ru} - f_{rv} + f_{sv} - f_{su}) \cdot (d_{\pi(s)\pi(u)} - d_{\pi(s)\pi(v)}) \\ & + d_{\pi(r)\pi(v)} - d_{\pi(r)\pi(u)} (f_{ur} - f_{vr} + f_{vs} - f_{us}) \end{aligned}$$

$$\cdot (d_{\pi(u)\pi(s)} - d_{\pi(v)\pi(s)} + d_{\pi(v)\pi(r)} - d_{\pi(u)\pi(r)}). \quad (13.3.17)$$

以上基于邻域最简单的局部搜索算法是重复改进法 (iterative improvement). 此处以 2-opt 为代表. 重复改进法有两种实现方式, 分别是: (1) 首次改进方式 (first-improvement), 即将邻域中的每个解和当前解进行比较, 如果优于当前解, 则立即替换当前解, 直到完成邻域搜索; (2) 最优改进方式 (best-improvement), 即所有邻域先进行比较, 取邻域中最优解与当前解进行比较, 如果优于当前解则替换之. 最优改进方式的优点在于每次交换时, 可以利用前次迭代的信息, 其中第一次迭代的时间为 $O(n^3)$, 以后为 $O(n^2)$; 使用首次改进方式通常需要迭代更多次数后才到达局部最优, 且每次邻域扫描的时间为 $O(n^3)$, 但是, 如果只使用有限次迭代次数, 首次改进方式将会更快.

迭代改进算法的缺点是当其遇到第一个局部最优解时将停止运行. 可以通过禁忌搜索 (TS)、模拟退火 (SA) 使其跳出局部最优. 通常情况下, TS 和 SA 将会比仅仅使用 2-opt 得到更优的解质量, 但却需要花费更长的时间.

前面提到的求解 QAP 的蚁群算法使用了不同的局部搜索算法, 如在 AS-QAP 中, 用到最优改进 2-opt 与短时间的 SA; 在 ANTS-QAP 中只使用了最优 2-opt; 在 MMAS-QAP 中, 使用短时间的稳健禁忌搜索和最优 2-opt; HAS-QAP 与 FANT 使用首次改进 2-opt.

表 13-3 为求解 QAP 的蚁群算法与其他启发式算法的对比, 其中 SA 为模拟退火, TS 为禁忌搜索, GA 为遗传算法, AS-LS 为带局部搜索的蚂蚁系统, AS-SA 为蚂蚁系统与模拟退火的混合算法. 表中的黑体表示最优解.

表 13-3 蚁群算法与其他算法比较 (解 QAP)

问题实例	SA	TS	GA	AS-QAP	AS-LS	AS-SA
Nugent(7)	148	148	148	148	148	148
Nugent(12)	578	578	588	578	578	578
Nugent(15)	1150	1150	1160	1150	1150	1150
Nugent(20)	2570	2570	2688	2598	2570	2570
Nugent(30)	6128	6124	6784	6232	6146	6128
Elshafei(19)	17937024	17212548	17640584	18122850	17212548	17212548
Krarup(30)	89800	90090	108830	92490	89300	88900

从表 13-1 中可以看出, 在求解 QAP 时, AS-QAP 算法的性能优于遗传算法, 但比模拟退火和禁忌搜索要差; 当在蚁群算法中引入局部搜索 (如 AS-LS), 和模拟退火 (如 AS-SA) 以后, 蚁群算法的性能得到明显地提高, 与 SA, TS 具有相当的性能. 顺便指出, 蚁群算法 MMAS 与局部搜索结合以后, 是目前求解 QAP 最好的算法.

第四节 蚁群优化算法与车间作业调度问题

一、车间作业调度问题

(一) 调度问题

所谓调度 (Scheduling), 就是为了实现某一目的而对共同使用的资源实行时间分配. 如车间作业调度问题 (job-shop scheduling problem, JSP) 就是为了处理多项不同的事务而如何分配作为资源的机械设备, 并使总的作业时间最少的问题. JSP 是 NP 完全问题中最难的问题之一, 即使 JSP 中比较简单的流动车间作业问题 (flow shop scheduling problem, FSSP) 也是与非对称旅行商问题 (traveling salesman problem, TSP) 难度相当的一类问题.

以往, 求解 JSP 的主要方法是分支定界 (branch and bound, BAB). 近年来一些学者使用比较好的求解方法如神经网络、模拟退火、遗传算法、蚂蚁算法等. 本节将讨论蚂蚁算法在 JSP 中的应用.

(二) 作业车间调度问题

考虑 n 项事务在 m 台设备上处理的问题, 其中, 假设每台设备不能同时处理两项以上的事务, 各事务也不能在两项以上的设备上同时处理. 我们把每台设备上各种事务的处理称为作业, 把每项事务所需要的设备使用顺序称为技术顺序. 各作业的处理时间是预先给定的.

一般地, $m \times n$ 调度问题可以有以下三种分类方法:

(1) 根据设备环境分类

1) 开放式车间 (O): 不特别给定某一技术顺序; 2) 流动车间 (F): 给定技术顺序但不能预知事务; 3) 作业车间 (J): 给定技术顺序和事务但每项事务都不相同, 等等;

(2) 根据事务特点分类

1) 允许作业中断 (P_{pmtn}); 2) 作业处理时间全相等 (P_u); 3) 处理时间不限制 (G), 等等;

(3) 根据某一最佳标准分类

1) 总作业时间 (C_{\max}) 最小; 2) 总延迟 (L_{\max}) 最小, 等等.

如果采用以上的分类方法, 则总作业时间最小的 JSP 可表示为 $J|G|C_{\max}$ 类型的问题.

二、蚂蚁算法与车间作业调度问题

为了用蚂蚁算法求解 JSP, 记机器集合为 M , 作业集合为 J . 作业 $j(j = 1, 2, \dots,$

J) 包含一系列属于全部操作集 $O = \{\dots o_{jm} \dots\}$ 的固定顺序的操作. $N = |O|$ 是操作的总数. 操作 $o_{jm} \in O$ 属于作业 j , 且必须在机器 m 上进行加工, 需要连续加工的时间为 d_{jm} . 则 JSP 即将所有作业的操作分配给相应的时间间隔, 使得没有两个作业在同一时刻放在同一台机器上处理, 完成所有操作的时间最小.

对于有 M 台机器, J 个作业和总操作数为 N 的操作集 O 的 JSP 可以用一个有向带权图 $Q = (O', A)$ 来表示. 其中 $O' = O \cup \{o_0\}$, A 是所有弧的集合, A 由连接节点 o_0 到所有作业的第一个操作的有向弧和集合 O 中除了属于同一个作业的操作 (节点) 的全连接弧组成. 属于同一作业的节点以顺序的方式连接, 即每个节点只与其直接后继节点相连.

当几个作业的第一个操作在同一台机器上加工时, 节点 o_0 在决定哪个作业首先被调度必不可少. 于是在图 Q 中共有 $N+1$ 个节点和 $\frac{N(N+1)}{2} + |J|$ 条弧. 除节点 o_0 外, 所有的节点都是成对连接的. 节点 o_0 只连接到每个作业的第一个操作. 与每条弧相关有一对数 $\{\tau_{kl}, \eta_{kl}\}$, 其中 τ_{kl} 是信息素的迹, 而 η_{kl} 是期望程度 (即启发式因子). η_{kl} 按与问题相关的某种启发式信息, 如最长处理时间、最短完成时间等来确定. 蚂蚁所走过的节点即构成了问题的一个解. 如考虑 3×2 (3 道作业, 2 台机器) 的 JSP, 则可以将其表示成图 13-6 的形式. 此处假设第一台机器处理操作 1, 3, 5, 剩下的操作由第二台机器处理.

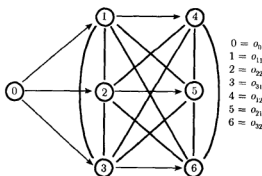


图 13-6 AS 算法求解 3 作业、2 机器 JSP 的图形表示 (其中的粗线表示一对有向弧)

初始时刻, 将所有的蚂蚁置于节点 o_0 ; 其后, 每只蚂蚁必须对下一步移动到那个剩下的节点做出选择. 为处理该问题, 蚂蚁使用的转移概率公式与 AS 算法中略有不同. 为了使蚂蚁行走时产生可行解, 在每个节点除了定义一个禁忌表外, 还要定义一个允许到达的节点集. 如对于每只蚂蚁 k , 设节点集 G_k 是蚂蚁 k 将要访问的节点集, S_k 是下一步允许访问的节点集, 则转移概率公式为

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{s \in allowed_k} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}(t)]^\beta}, & \text{如果 } j \in S_k, \\ 0, & \text{否则.} \end{cases} \quad (13.4.1)$$

当某个节点被选择后,该节点将会加入蚂蚁 k 的禁忌表 $\text{tabu}(k)$,同时从 G_k 和 S_k 中删除该节点;当被选择的节点不是作业的最后操作,则该作业的直接后继节点将会加入到 S_k 中.该过程可确保蚂蚁产生的解为可行解.当 $G_k = \emptyset$ 时,迭代过程结束,则蚂蚁 k 的禁忌表中的节点顺序即为问题的一个解.假如某只蚂蚁所经过的节点顺序为 $\pi = (0, 1, 4, 2, 5, 3, 6)$,则这些节点对应的操作顺序(解)为 $\{(1, 5), (1, 3), (5, 3)\}$ 和 $\{(4, 2), (4, 6), (2, 6)\}$.

当所有蚂蚁都找到问题的解之后,蚂蚁经过的路径上的信息素需要进行更新,其更新方式与 ant-cycle 算法相同.

该算法在求解 $10 \times 10, 10 \times 15$ JSP 时,获得了较好的结果^[25].

第五节 蚁群优化算法与网络路由问题

一、网络路由问题及算法

网络路由算法的目的是生成从源节点到目的节点的路由,同时使得网络的性能最大,通讯费用最小.从这个意义上说,网络路由问题是一种受网络交换、传输等约束条件限制的动态、随机多目标规划问题.通常考虑的网络性能是指网络吞吐量(throughput)和数据报的平均时延(average delay).前者是对网络在某一段时间内能够提供的服务数的量化,而后者定义了相同的时间内网络所能提供的服务质量.

路由算法可分为静态路由和动态路由两大类.静态路由算法不考虑网络的当前状态,数据报的路由信息在源节点就已确定,该路由通常按照某种最小费用(如最短路径)进行选择;动态路由通常更具优势,因为它总是试图适应网络通讯量的不断变化状态而产生相应的路由策略.当然这种自适应路由算法也有一定的缺陷如容易产生震荡,从而导致一些性能的不稳定.

二、简单蚂蚁网

在简单蚂蚁网(simple ant net, S-AntNet)中,每只蚂蚁在源节点与目的节点之间搜索最小费用的路径.蚂蚁从源节点一步一步向目的节点移动,当蚂蚁 k 在节点 i 时,按概率决策规则移动到下一个节点 j .概率决策规则是蚂蚁 k 的内存信息 M^k 和局部蚂蚁路由表 A_i 的函数.

S-AntNet 中信息素的迹与网络中的弧相关联,但存储在与弧-目的相关的变量中,即对每条有向弧 (i, j) 与 $N_c - 1$ 个信息素的值 $\tau_{ijd} \in [0, 1]$ 相关联.每条弧

也有一个与目的节点无关的启发式值 $\eta_{ij} \in [0, 1]$, 其定义如下:

$$\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{l \in N_i} q_{il}}, \quad (13.5.1)$$

其中 q_{ij} 是节点 i 与其邻域节点 j 之间链路上的队列长度 (等待被发送数据的长度). 与大多数其他路由问题的 ACO 实现一样, S-AntNet 中没有使用 daemon 操作.

局部蚂蚁路由表 A_i 由局部信息素的迹 τ_{ija} 与启发式因子 η_{ij} 组合而成. 蚂蚁在生成到目的节点路径的过程中和数据报使用同样的链路队列, 因此蚂蚁与数据报的时延相同, 其值真实地反应出当前该路径上网络的状况, 从而可用其作为衡量该路径好坏的依据.

一旦蚂蚁 k 生成了一条路径, 便在其走过的节点上释放一定数量的信息素 $\Delta\tau^k(t)$, 其数量与生成路径的好坏成正比. 当蚂蚁到达目的节点后, 沿原路返回到源节点时, 采用较高优先级队列, 以使收集到的信息能尽快地反传.

算法 13.1 是 S-AntNet 中蚂蚁的活动过程.

procedure new_active_ant(ant_identifier)

$k = \text{ant_identifier}; I = \text{get_start_node}(); t = \text{get_end_node}(); s^k(t) = s_s^k;$

$M^k(t) = i;$

while ($i \neq t$)

foreach $j \in N_i^k$ **do** read (a_{ij});

foreach $j \in N_i^k$ **do** $[P]_{ij} = p_{ij} = \frac{a_{ij}}{\sum_{l \in N_i^k} a_{il}};$

 next_node = apply_probabilistic_rule(P, N_i^k);

$i = \text{next_node}; s^k(t) = \langle s^k(t), i \rangle;$

$M^k(t) = i;$

while end

foreach $l_{ij} \in \psi^k(t)$ **do**

$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau^k(t);$

$a_{ij}(t) \leftarrow \frac{w \cdot \tau_{ij}(t) + (1-w)\eta_{ij}}{w + (1-w) \cdot (|N_i^k| - 1)}$

end foreach

 free_all_allocated_resources();

end procedure

当蚂蚁 k 到达目的节点后, 将沿原路返回, 同时更新其经过的每个连接 l_{ij} 上的信息素的迹, 即

$$\tau_{ija}(t) \leftarrow \tau_{ija}(t) + \Delta\tau^k(t). \quad (13.5.2)$$

之所以让蚂蚁在往回走的过程中更新信息素的迹,是因为只有当蚂蚁到达目的节点之后才能计算 $\Delta\tau^k(t)$ 的值。

等所有信息素更新完后,与节点 i 、邻域节点 j 和目的节点 d 相关的信息素更新为

$$\tau_{ijd}(t) \leftarrow \frac{\tau_{ijd}(t)}{(1 + \Delta\tau^k(t))}, \forall j \in N_i, \quad (13.5.3)$$

其中, N_i 为节点 i 的邻域集合。

S-AntNet 中节点 i 的路由表可表示为

$$a_{ijd}(t) = \frac{w \cdot \tau_{ijd}(t) + (1-w) \cdot \eta_{ijd}}{w + (1-w) \cdot (|N_i| - 1)}, \quad (13.5.4)$$

其中 $j \in N_i$, d 为目的节点, $w \in [0, 1]$ 为权重因子。

蚂蚁的决策规则如下:在时刻 t , 设蚂蚁 k 在节点 i , 其目的节点为 d 。如果 $N_i \not\subset M^k$, 则在蚂蚁的当前邻域中至少有一个节点蚂蚁还没有访问, 则蚂蚁按如下规律选择下一个节点 j :

$$P_{ijd}^k(t) = \begin{cases} a_{ijd}(t), & \text{if } j \notin M^k, \\ 0, & \text{if } j \in M^k, \end{cases} \quad (13.5.5)$$

否则, 蚂蚁按一致的概率 $P_{ijd}^k(t) = 1/(|N_i|)$ 选择下一个节点 j 。

换句话说, 蚂蚁尽量避免产生环路, 但是当节点 i 的所有邻域都被访问以后, 蚂蚁不得不访问已访问过的节点, 这样就会产生一个环路, 该环路将从蚂蚁的内存中删除。考虑到蚂蚁决策路径的随机性和通讯状态的进化过程, 蚂蚁不可能一直重复该环路。

第六节 蚁群算法与 0-1 背包问题

背包问题 (knapsack problem, KP)^[5,26] 是一个经典的组合优化问题, 有着广泛的实际应用背景, 如预算控制、项目选择、材料切割和货物装载等。背包问题可以描述为: 给定 n 件物品和一个背包, 物品 $i (i = 1, 2, \dots, n)$ 的体积是 w_i , 其价值为 p_i , 背包的容量为 V 。问应如何选择装入背包的物品, 使得装入背包中物品的总价值最大?

背包问题是一个 NP- 难题^[5], 目前已有的求解方法分为精确算法 (如动态规划、回溯法和分支定界等指数级方法) 和近似算法 (如贪婪算法、Lagrange 法、遗传算法等) 两大类。下面将给出一种基于蚁群优化思想的新型近似算法。

一、求解背包问题的蚁群优化算法

(一) 0-1 背包问题的图形表示

图 13-7 为 0-1 背包问题的构造图。该图由 $n+1$ 个节点按先后顺序排列而成。

从节点 $i(i = 1, 2, \dots, n)$ 出发共有 n 条有向线段 $a[i, j](j = 1, 2, \dots, n)$ 连接到节点 $i+1$ 。在 $a[i, j](i, j = 1, 2, \dots, n)$ 上有两个数 w_j (第 j 个物品的体积) 和 p_j (第 j 个物品的价值) 与其相关联。

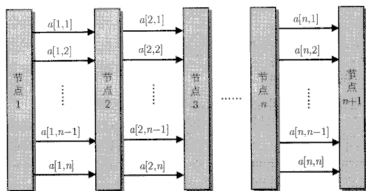


图 13-7 0-1 背包问题的构造图

(二) 蚂蚁的路径选择

设 $\tau_{ij}(t)$ 为 $t(t = 0, 1, 2, \dots)$ 时刻有向线段 $a[i, j]$ 上的信息素, 初始时刻各有向线段上的信息素为 $\tau_{ij}(0) = C$ (C 为较小的正常数)。在时刻 t , 将生成的 m 只蚂蚁放到节点 1 上, 然后每只蚂蚁根据路径上的信息素和启发式因子独立地选择某一条有向线段并移动到下一个节点, 直到不能向前移动为止。在时刻 t , 蚂蚁 $k(k = 1, 2, \dots, m)$ 从节点 $i(i = 1, 2, \dots, n)$ 经由线段 $a[i, j]$ 转移到节点 $i+1$ 的转移概率 $p_{ij}^k(t)$ 为

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{s \in J_k(i)} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}]^\beta}, & \text{如果 } j \in J_k(i), \\ 0, & \text{否则,} \end{cases} \quad (13.6.1)$$

其中:

η_{ij} ——蚂蚁选择有向线段 $a[i, j]$ 的期望值, $\eta_{ij} = p_j/w_j$. p_j 、 w_j 分别为物品 j 的价值和体积. η_{ij} 越大, 选择物品 j 的可能性越大;

$J_k(i)$ ——位于节点 i 的蚂蚁 k 当前能够选择的有向线段集合, $J_k(i) = \{1, 2, \dots, n\} - \text{tabu}_k$. 蚂蚁 k 的禁忌表 tabu_k 记录其当前走过的有向线段。在图 13-7 中, 对有向线段 $a_1[i_1, j_1]$ 和 $a_2[i_2, j_2]$, 如果 $j_1 = j_2$, 则认为是相同的有向线段。蚂蚁 k 在向前移动的过程中不允许选择相同的有向线段, 该过程由禁忌表 tabu_k 控制。

α 和 β ——分别表示蚂蚁 k 在选择路径时, 有向线段上信息素和启发式因子的相对重要程度。

与蚂蚁系统不同, ACA-KP 中还增加了另外一种选择路径的方式. 当按式 (13.6.1) 选择的有向线段不满足 0-1 背包问题的约束条件时, 构造集合 $\text{Sub-}J_k(i)$, 且使其满足: ① $\text{Sub-}J_k(i) \subseteq J_k(i)$; ② $\text{Sub-}J_k(i)$ 中的物品加入背包后能满足约束条件. 如果 $\text{Sub-}J_k(i)$ 非空, 则按式 (13.6.2) 选择:

$$m = \arg \max_{j \in \text{Sub-}J_k(i)} (p_j), \quad (13.6.2)$$

$a[i, m]$ 即为选择的有向线段.

蚂蚁 k 在选择有向线段时, 公式 (13.6.1) 具有较高的优先级. 即在选择路径时, 首先利用公式 (13.6.1) 进行选择, 如果公式 (13.6.1) 选择的有向线段不满足背包问题的约束条件, 再按式 (13.6.2) 进行选择. 该过程将一直持续到蚂蚁 k 不能再选择到满足约束条件的有向线段为止. 此时, 蚂蚁 k 将自动死亡. 假如蚂蚁 k 死亡后禁忌表 tabu_k 中的数字为 $\{j_1, j_2, \dots, j_r\}$ (其中 r 为蚂蚁 k 走过的有向线段数, 且 $\{j_1, j_2, \dots, j_r\} \subseteq \{1, 2, \dots, n\}$), 则蚂蚁 k 求得背包问题的解为

$$L_k = p_{j_1} + p_{j_2} + \dots + p_{j_r}. \quad (13.6.3)$$

(三) 信息素更新

当所有 m 只蚂蚁都死亡以后, 可求得 m 组可行解. 如果本次迭代最好解优于当前最好解, 则用其替换当前最好解. 之后, 蚂蚁 k 要对其经过的路径上的信息素进行更新, 即

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}, \quad \rho \in (0, 1), \quad (13.6.4)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (13.6.5)$$

其中 ρ 表示路径上信息素的蒸发系数, $\Delta\tau_{ij}$ 表示本次迭代中有向线段 $a[i, j]$ 上信息素量的增量, $\Delta\tau_{ij}^k$ 表示第 k 只蚂蚁在 $a[i, j]$ 上留下的信息素量, $\Delta\tau_{ij}^k$ 表示为

$$\Delta\tau_{ij}^k = \begin{cases} \frac{L_k}{Q}, & \text{若蚂蚁 } k \text{ 的本次路径包含 } a[i, j], \\ 0, & \text{否则,} \end{cases} \quad (13.6.6)$$

其中, Q 为正常数, L_k 表示第 k 只蚂蚁在本次迭代中所求得解.

(四) ACA-KP 算法描述

ACA-KP 算法的实现步骤为:

Step 1 初始化参数: $\alpha, \beta, \rho, Q, NC_{\max}, m, \tau_{ij}(0)$, 其中 $i, j = 1, 2, \dots, n$;

Step 2 生成 m 只蚂蚁, 并将其置于节点 1;

Step 3 for 每只蚂蚁 do

- 1) 按公式 (13.6.1) 计算转移概率并选择下一条有向线段;
- 2) 如果选择的有向线段不满背包问题的约束条件, 则按公式 (13.6.2) 重新选择, 如果仍不满足背包问题的约束条件, 则该蚂蚁死亡;
- 3) 如果蚂蚁未死亡, 则将所选择有向线段的序号加入蚂蚁 k 的禁忌表 $tabu_k$.

end for

Step 4 计算本次迭代的最好解, 如果其优于当前最好解, 则用其替换当前最好解;

Step 5 按公式 (13.6.4) 更新路径上的信息素;

Step 6 if NC 小于 NC_{\max} 且未进入停滞状态 then

- 1) 清空所有蚂蚁禁忌表中的数据;
- 2) $\Delta\tau_{ij} := 0$;
- 3) $NC := NC + 1$;
- 4) 转至 Step2.

else

输出最优解;

end if

二、实验结果

(一) ACA-KP 算法的参数设置

ACA-KP 与蚂蚁系统的差别在于: (1) 问题的图形表示不同; (2) 状态转移策略不同; (3) ACA-KP 中的蚂蚁在没有走完所有节点时就死去. 为了测试各参数对算法的影响, 首先通过一个中等规模的背包问题 (物品数 $n = 60$) 对其参数进行研究.

参数的测试值为: $\alpha = \{0.1, 0.2, 0.5, 1.0, 2.0, 3.0\}$, $\beta = \{1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 15.0\}$, $\rho = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $Q = \{1.0, 10.0, 100.0, 1000.0, 10000.0\}$. 在测试某个参数时, 其他参数取缺省值 (各参数的缺省值为: $\alpha = 1.0, \beta = 2.0, \rho = 0.5, Q = 10.0$). 对参数的某一取值, 运行 ACA-KP 算法 15 次. 表 13-4、表 13-5、表 13-6、表 13-7 中的 Avg 为 15 次结果的平均值, Best 为 15 次中的最好解, Error(%) 的计算方法为:

$$\text{Error}(\%) = (\text{算法所发现的最好解} - \text{本次测试的平均值}) \times 100 / \text{算法所发现的最好解}.$$

其中, 算法发现的最好解为 12390, Error(%) 值越小, 说明参数的值越好, 反之则越差.

表 13-4 参数 Q 对 ACA-KP 的影响

Q	Avg	Error(%)	Best
1	12346.4	0.35	12387
10	12353.2	0.30	12387
100	12351	0.31	12387
1000	12351.8	0.30	12374
10000	12331.4	0.47	12387

表 13-5 参数 α 对 ACA-KP 的影响

α	Avg	Error(%)	Best
0.1	12385.7	0.03	12387
0.2	12387.3	0.02	12390
0.5	12379.2	0.09	12387
1.0	12353.2	0.30	12387
2.0	12270.2	0.97	12355
3.0	12288.0	0.82	12374

表 13-6 参数 β 对 ACA-KP 的影响

β	Avg	Error(%)	Best
1.0	12353.2	0.30	12387
2.0	12385.7	0.03	12387
3.0	12388.5	0.01	12390
4.0	12388.5	0.01	12390
5.0	12389.7	0.002	12390
6.0	12390.0	0.00	12390
7.0	12388.8	0.01	12390
8.0	12390.0	0.00	12390
9.0	12388.8	0.01	12390
10.0	12387.0	0.02	12387
15.0	12387.0	0.02	12387

表 13-7 参数 ρ 对 ACA-KP 的影响

ρ	Avg	Error(%)	Best
0.1	12376.1	0.11	12387
0.2	12378.3	0.09	12390
0.3	12351.2	0.31	12387
0.4	12351.2	0.31	12390
0.5	12353.2	0.30	12387
0.6	12349	0.33	12390
0.7	12341.8	0.39	12387
0.8	12311.2	0.64	12387
0.9	12328.2	0.50	12390

从表 13-4 可以看出, Q 对算法的影响不大, 该结论与文献 [25] 中的结论一致; 表 13-5 显示当 $\alpha = \{0.1, 0.2\}$ 时, ACA-KP 有较高的性能, 与文献 [25] 中的结论有较大的差别; 表 13-6 显示 $\beta = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 15\}$ 算法有较高的性能, 与文献 [25] 不同; 表 13-7 显示 $\rho = \{0.1, 0.2\}$ 时有利于发现较好的解。

(二) ACA-KP 求解文献 [26] 中的背包问题

首先用 ACA-KP 求解文献 [26] 中的例 2. 参数为 $\alpha = 0.2, \beta = 5.0, \rho = 0.2, Q = 10.0$, 蚂蚁数为 10, 迭代次数为 100. 实验发现, ACA-KP 迭代 3~8 代后总能找到最优解 1024, 而文献 [26] 需要迭代 200 代。

(三) ACA-KP 与文献 [27] 中 Ar[2] 算法的比较

通过随机生成各种不同规模的 0-1 背包问题实例, 利用 ACA-KP 与文献 [27] 中的 Ar[2] 算法 (修复遗传算法) 进行对比实验. Ar[2] 的群体大小为 500, 变异和交叉概率分别 0.05 和 0.65, 采用 5% 的修正规则。

对各种规模 ($n=50, 100, 200, 500, 1000, 2000$), 按物品的体积 w_i 和价值 p_i 生成三种不同范围 ($R1=[1, 10], R2=[1, 100], R3=[1, 500]$) 的背包问题实例. 如当 $n=100$ 时, 将分别生成 $w_i, p_j \in R1 (i=1, 2, 3)$ 三种类型的背包实例. 背包容积为 $V = \frac{1}{2} \sum_{i=1}^n w_i$.

表 13-8 为 ACA-KP 与 Ar[2] 求解各问题的比较. 对于 $n=50, 100, 200$ 的背包问题实例, ACA-KP 和 Ar[2] 分别运行 15 次后的最好解的平均值; 对 $n=500, 1000, 2000$ 时, 算法 ACA-KP 和 Ar[2] 各运行 5 次后最好解的平均值. 表 13-8 中的黑体表示较好的解. 从表 13-8 中可以看出, ACA-KP 发现较好解的可能性较大. 实验中还发现, 在运行相同代数情况下, ACA-KP 所需的时间比 Ar[2] 要长。

表 13-8 ACA-KP 与 Ar[2] 求解各问题的比较

N	迭代次数	R1 = [1, 10]		R2 = [1, 100]		R3 = [1, 500]	
		ACA-KP	GA	ACA-KP	GA	ACA-KP	GA
50	500	206	204	2296	2276	9994	9994
100	500	376	380	4096	4085	19932	19886
200	500	629	632	7263	7298	42368	42062
500	1000	1812	1802	31541	30412	186524	185864
1000	1500	4651	4669	60125	59625	356528	356253
2000	1500	11205	11159	142563	142651	952368	952035

ACA-KP 算法能很好地求解 0-1 背包问题. 该算法首先将 0-1 背包问题转化为相应的带权图, 并针对该带权图设计了两个状态转移公式. 蚂蚁分别根据这两个状态转移公式在带权图中移动直到死亡. 此时, 蚂蚁走过的路径即为 0-1 背包问题的一个解. 实验对 ACA-KP 算法的参数进行了研究, 发现其与 AS 算法 (求解 TSP)

的最优参数集有较大的差异,其原因可能是求解 KP 和 TSP 的蚁群优化算法的图形表示和路径选择有较大的差异.由此可见,蚁群优化算法的最优参数集会随待求解问题的不同而有所不同.在与文献 [26] 中的算法进行比较时发现,ACA-KP 算法具有更高的性能.对大规模 0-1 背包问题的求解发现,蚁群优化算法较 Ar[2] 算法有较高的性能,但需要更长的计算时间.为了缩短 ACA-KP 算法的求解时间,可考虑将公式 (13.6.1) 修改成蚁群系统 [28] 中的伪随机比例选择方式;另外,并行实现也是一个很好的途径.

第七节 蚁群优化算法与三维空间机器人路径规划

三维空间机器人路径规划的任务是在具有障碍物的环境中,按照一定的评价标准,寻找一条从起始位置到目的位置的无碰路径.解决该问题的常规方法由两部分组成:(1) 建立一个数据结构来代表工作空间的几何结构;(2) 搜索该数据结构以找到一条无碰路径,如可视顶点图法 (visibility graphic) 和 voronoi 图法等.但 these 方法需要大量的计算时间来建立和搜索数据结构,因此不适用于存在运动障碍物的在线路径规划.下面将蚁群算法应用于该问题,实验取得了较好的结果.

一、三维空间有效路径的表示

如图 13-8 所示,在笛卡尔坐标系 $O-XYZ$ 下,设机器人所在的位置为 S ,要到达的目的点为 D (设 S 到 D 的长度为 h),障碍物 O_1, O_2, \dots, O_k 在 S 与 D 之间,则路径规划的目的就是要在 S 和 D 之间找到一条既短又安全的路径.

不妨设障碍物为球形,则可将其表示为 $\{(o_i, r_i) | i = 1, 2, \dots, k\}$, 其中 o_i 、 r_i 分别表示第 i 个障碍物的球心位置和半径长度.

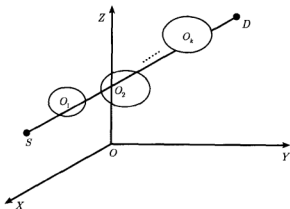


图 13-8 空间障碍物在笛卡尔坐标系 $O-XYZ$ 中

为了研究的方便,建立图 13-9 所示的笛卡儿坐标系 $O'-X'Y'Z'$. 其中 S 点为新坐标系的原点, SD 方向为 Z' 轴的正方向, X' 轴和 Y' 轴可适当选择.

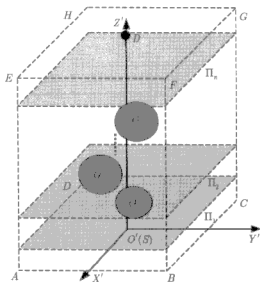


图 13-9 空间障碍物在坐标系 $O'-X'Y'Z'$ 中

坐标系 $O'-X'Y'Z'$ 与 $O-XYZ$ 之间的变换关系为

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos \alpha_x & \cos \alpha_y & \cos \alpha_z \\ \cos \beta_x & \cos \beta_y & \cos \beta_z \\ \cos \gamma_x & \cos \gamma_y & \cos \gamma_z \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, \quad (13.7.1)$$

其中 $\alpha_x, \beta_x, \gamma_x$ 分别为 X 轴与 X', Y', Z' 轴的夹角; $\alpha_y, \beta_y, \gamma_y$ 分别为 Y 轴与 X', Y', Z' 轴夹角; $\alpha_z, \beta_z, \gamma_z$ 分别为 Z 轴与 X', Y', Z' 轴夹角.

由公式 (13.7.1) 可计算出障碍物 $\{(o_i, r_i) | i = 1, 2, \dots, k\}$ 在坐标系 $O'-X'Y'Z'$ 下的坐标, 设为 $\{(o'_i, r'_i) | i = 1, 2, \dots, k\}$. 由 SD 的长度为 h , 知 D 在坐标系 $O'-X'Y'Z'$ 下的坐标为 $(0, 0, h)$.

如图 13-9 所示, 在坐标系 $O'-X'Y'Z'$ 中作立方体 $ABCDEFGH$, 其中立方体的 $ABCE$ 面在 $X'Y'$ 平面上, 且为边长为 $2L$ 的正方形, 边 AB 平行于 Y' 轴, BC 平行于 X' 轴, 且原点 O' 在正方形 $ABCD$ 的中心上, 立方体的高 $AE = h$, 定 点 D 的坐标为 $(-L, -L, 0)$.

将 $O'D$ 进行 $(n+1)$ 等分, 过每个等分点, 作垂直于 Z' 轴的 n 个平面 $\Pi_i (i = 1, 2, \dots, n)$. 平面 Π_i 与立方体 $ABCDEFGH$ 相交为图 13-10 所示的正方形 (为方便起见, 记为 Π_i). 如图 13-10 所示将正方形 $\Pi_i (i = 1, 2, \dots, n)$ 平均分成 $m \times m$ 个

小正方形. 则对于正方形 Π_i 上小正方形顶点 $P^i(u, v) (u, v = 0, 1, \dots, m)$ 在坐标系 $O' - X'Y'Z'$ 中的实际坐标为 $\left(-L + \frac{u * 2 * L}{m}, -L + \frac{v * 2 * L}{m}, \frac{i * h}{n}\right)$.

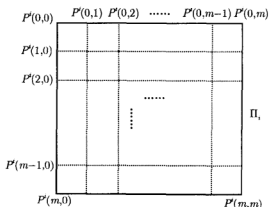


图 13-10 平面 Π_i 上网格坐标表示

二、基于蚁群算法的路径规划

有效路径的生成: 在 $O' - X'Y'Z'$ 坐标系中, 蚁群从 S (原点) 出发, 首先到达平面 Π_1 上的某点 $p^1(i_1, j_1)$, 其中 $i_1, j_1 \in \{0, 1, 2, \dots, m\}$, 然后再从 $p^1(i_1, j_1)$ 点出发到达平面 Π_2 上的某点 $p^2(i_2, j_2)$, 其中 $i_2, j_2 \in \{0, 1, 2, \dots, m\}$, ..., 最后到达平面 Π_n 上的 $p^n(i_n, j_n)$ 点, 其中 $i_n, j_n \in \{0, 1, 2, \dots, m\}$. 连接 $p^n(i_n, j_n)$ 与 D 点, 便构成了一条从 S 到 D 的路径 $S \rightarrow p^1(i_1, j_1) \rightarrow \dots \rightarrow p^n(i_n, j_n) \rightarrow D$. 由于 S 与 D 之间存在障碍物, 因此某些路径可能是无效的. 为了让蚂蚁尽量生成有效路径, 此处为平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上的点 $p^i(u, v)$ 引入一个允许列表 $allowed^i(u, v)$. $allowed^i(u, v)$ 为平面 Π_{i+1} 上点集的子集, 且连接 $p^i(u, v)$ 与 $allowed^i(u, v)$ 中的任何点都不会穿过障碍物.

(1) 允许列表 $allowed^i(u, v)$ 的计算方法

设 $p^i(u, v)$ 为平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上的某一点, 现计算该点的 $allowed^i(u, v)$ 列表. 对平面 Π_{i+1} 上的任意点 $p^{i+1}(k, l) (k, l = 0, 1, \dots, m)$, 如果线段 $p^i(u, v)p^{i+1}(k, l)$ 不与任何障碍物相交, 则将 $p^{i+1}(k, l)$ 点加入到 $allowed^i(u, v)$ 中. 按照此方法, 可以计算出点 $p^i(u, v)$ 的所有允许到达的点, 并将其存于 $allowed^i(u, v)$ 中.

(2) 信息素列表

对于平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上的任意点 $p^i(u, v)$, 有一个信息素列表 τ_{uv}^i 与之相对应. 信息素列表 τ_{uv}^i 是一个 $(m+1) \times (m+1)$ 数组, 其中 $\tau_{uv}^i(k, l)$ 表示点 $p^i(u, v)$ 与点 $p^{i+1}(k, l)$ 之间的信息素连接强度.

(3) 选择概率

对于处于平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上的任意点 $p^i(u, v)$ 的蚂蚁, 选择平面 Π_{i+1} 上的 $p^{i+1}(k, l)$ 的概率为

$$P = \frac{[\tau_{uv}^i(k, l)]^\alpha}{\sum_{p^{i+1}(x, y) \in allowed^i(u, v)} [\tau_{uv}^i(x, y)]^\alpha}, \quad (13.7.2)$$

其中 α 表示信息素的重要程度.

(4) 信息素更新

对于到达目的点 D 的蚂蚁, 最优解蚂蚁所经过的路径上的信息素将会按公式 (13.7.3) 进行更新,

$$\tau_{uv}^i(k, l) = (1 - \rho) * \tau_{uv}^i(k, l) + \Delta \tau_{uv}^i(k, l), \quad \rho \in (0, 1), \quad (13.7.3)$$

其中 ρ 表示信息素的蒸发系数,

$$\Delta \tau_{uv}^i(k, l) = Q/L, \quad (13.7.4)$$

Q 为常数, L 为最优解蚂蚁经过路径的长度.

于是蚁群算法的流程为

Step 1 初始化

(1) 按公式 (13.7.1) 计算出各障碍物在坐标系 $O' - X'Y'Z'$ 中的坐标;

(2) 作立方体 $ABCDEFGH$ 和平面 $\Pi_i (i = 1, 2, \dots, n)$, 并将每个平面划分成 $m \times m$ 个小方格, 计算所有小方格顶点处的坐标;

(3) 计算平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上所有点的允许列表 $allowed^i(u, v) (u, v = 0, 1, \dots, m)$;

(4) 初始化平面 $\Pi_i (i = 1, 2, \dots, n-1)$ 上所有点 $p^i(u, v) (u, v = 0, 1, \dots, m)$ 的信息素列表 τ_{uv}^i ;

(5) 将 m 只蚂蚁放到 $O' - X'Y'Z'$ 的坐标原点, $NC = 0$.

Step 2 对蚂蚁 $i (i = 1, 2, \dots, m)$, 检查其是否到达目的点 D , 如果没有到达, 则检查蚂蚁 i 所在点的允许列表 $allowed^i(u, v) (u, v = 0, 1, \dots, m)$, 如果允许列表不为空, 则按公式 (13.7.2) 独立地选择下一个平面上的点, 否则该蚂蚁死亡;

Step 3 对于能够到达目的点 D 的蚂蚁, 对其经过路径上的信息素进行全局更新;

Step 4 替换当前最优解;

Step 5 如果停机条件满足, 则输出最优解; 否则转步骤 2.

三、仿真实验结果

考虑到坐标系 $O-XYZ$ 与 $O'-X'Y'Z'$ 可以通过公式 (13.7.1) 相互转换, 本实验直接考虑坐标系 $O'-X'Y'Z'$ 中的情形. 设有 3 个障碍物 O_1, O_2, O_3 的球心坐标和半径分别为 $\{(20, 25, 60): 30\}$ 、 $\{(30, 25, 95): 40\}$ 、 $\{(46, 50, 260): 86\}$, 目的点 D 的坐标为 $(0, 0, 400)$. 蚁群算法的参数为 $\alpha = 2.0, \rho = 0.5, n = 15, m = 20, L = 150$. 用 Matlab 6.1 对仿真, 迭代 1200 次以后得最优解长度为 489.39, 如图 13-11 所示.

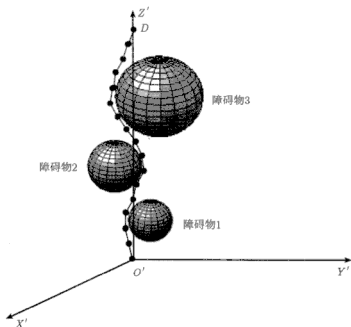


图 13-11 蚁群算法规划出的路径

同时还对该算法中的 m 、 n 进行了测试, 发现当 m 和 n 增大时, 算法可以得出更优的最优解, 但所需的计算时间将会增大.

蚁群算法在三维空间机器人路径规划问题的应用取得了较好的结果, 说明蚁群算法具有较强的鲁棒性. 该文采用将机器人所在位置与目的位置之间的空间按一定的规律划分成离散的点, 因此, 划分点的稠密程度决定了本算的精度. 离散点越稠密, 算法的精度越高, 所需时间也越多; 反之则越低, 所花时间越少. 在实际应用中, 可根据对精度的要求, 决定离散点的稠密程度.

第八节 本章小结

本章介绍了蚁群优化算法在 K-TSP、二次分配问题、车间作业调度问题、0-1

背包问题、机器人路径规划方面的应用。从蚁群优化算法的应用领域来看,该算法具有极强的推广能力。针对不同类型的问题,只需作较小的改变,就能应用于其他类型的问题。

参 考 文 献

- [1] Dorigo M, Maniezzo V, and Colomi A. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991
- [2] Maniezzo V, Colomi A, and Dorigo M. The Ant System Applied to the Quadratic Assignment Problem. Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994
- [3] Maniezzo V and Colomi A. The Ant System applied to the quadratic assignment problem. IEEE Transactions on Data and Knowledge Engineering, 1999, 11(5): 769~778
- [4] Colomi A, Dorigo M, Maniezzo V, and Trubian M. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science(JORBEL), 1994, 34: 39~53
- [5] 胡小兵, 黄席樾. 基于蚁群优化算法的 0-1 背包问题求解. 系统工程学报, 2005.6
- [6] Bullnheimer B, Hartl R F, and Strauss C. Applying the ant system to the vehicle routing problem. In Osman S Voß S, Osman I H, Martello S and Roucairol C, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academics, 1998. 109~120
- [7] Bullnheimer B, Hartl R F, and Strauss C. Applying the Ant System to the vehicle routing problem. In Martello S Voß S, Osman I H, and Roucairol C, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Dordrecht: Kluwer Academic Publishers, 1999. 285~296
- [8] Bullnheimer B, Hartl R F, and Strauss C. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 1999, 89: 319~328
- [9] Costa D and Hertz A. Ants can color graphs. Journal of the Operational Research Society, 1997, 48: 295~305
- [10] Di Caro G and Dorigo M. AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research, 1998, 9: 317~365
- [11] Bonabeau E, Henaux F, Guérin S, Snyers D, Kuntz P, and Theraulaz G. Routing in telecommunication networks with "Smart" ant-like agents[C]. In Proceedings of IATA'98, Second Int. Workshop on Intelligent Agents for Telecommunication Applications. Lectures Notes in AI vol. 1437, Springer Verlag, 1998
- [12] Di Caro G and Dorigo M. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997
- [13] Di Caro G and Dorigo M. Two ant colony algorithms for best-effort routing in datagram networks. In Pan Y, Akl S G, and Li K, editors. Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98). Anaheim: IASTED/ACTA Press, 1998. 541~546
- [14] Heusse M, Guérin S, Snyers D, and Kuntz P. Adaptive agent-driven routing and load balancing in communication networks. Advances in Complex Systems, 1998, 1(2-3): 237~254
- [15] Subramanian D, Druschel P, and Chen J. Ants and reinforcement learning: A case study in routing in dynamic networks. In Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997. 832~838

- [16] Van der Put R. Routing in the faxfactory using mobile agents. Technical Report R&D-SV-98-276, KPN Research, 1998
- [17] 吕国英, 刘泽民, 周正. 基于蚂蚁算法的分布式 QoS 路由选择算法. 通信学报, 2001, 22(9): 35~42
- [18] 张素兵, 刘泽民. 基于蚂蚁算法的分级 QoS 路由调度方法. 北京邮电大学学报, 2000, 23(4): 12~15
- [19] 张素兵, 刘泽民. 基于蚂蚁算法的时延受限分布式多播路由研究. 通信学报, 2001, 22(3): 71~74
- [20] 王颖, 谢剑英. 一种基于蚁群算法的多媒体网络多播路由算法. 上海交通大学学报, 2002, 36(4): 526~531
- [21] Lu Guoying, Liu Zemin. Qos Multicast Routing Based on Ant Algorithm in Internet. The Journal of China Universities of Posts and Telecommunication, 2000, 7(4): 12~17
- [22] 胡小兵, 黄席樾. 基于蚂蚁算法的三维空间机器人路径规划研究. 重庆大学学报(自然科学版), 2004, 26(8): 132~135
- [23] Frieze A M. An Extension of Christofides Heuristics to the k-Person Traveling Salesman Problem. Dis. Apply Math., 1983(6): 79~83
- [24] 王德荣, 刘方池. K-TSP 问题的近似算法. 华中理工大学学报, 2000, 28(8): 72~74
- [25] Dorigo M, Maniezzo V, and Colomi A. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29~41
- [26] 马良, 王龙德. 背包问题的蚂蚁优化算法. 计算机应用, 2001, 21(8): 4~5
- [27] 霍红卫, 许进, 保铮. 基于遗传算法的 0/1 背包问题求解. 西安电子科技大学学报, 1999, 26(4): 493~497
- [28] Dorigo M and Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53~66

第四部分 小样本统计学习

理论与支持向量机

第十四章 小样本统计学习的基本理论

第一节 引言

学习是一切智能系统最根本的特征,机器学习是人工智能最具智能特征、最前沿的研究领域之一。基于样本的机器学习问题是现代智能计算技术的一个重要分支,它模拟了人类从实例中学习归纳的能力,主要研究如何从一些观测数据(样本)中挖掘出目前尚不能通过原理分析得到的规律,并利用这些规律去分析客观对象,对未知数据或无法观测的新现象进行预测和判断。统计学在解决这类机器学习中起着基础性作用,但是,传统的统计学所研究的是渐近理论,即当样本趋向于无穷大时的极限特性。然而,现实应用当中,样本数目通常是有限的,有时样本的获取甚至是非常困难的,例如特殊故障模式下数据的获取就很困难。因此研究在小样本数据量下的统计学习规律是一个非常具有实用价值的问题。

以 V N Vapnik 为代表的学习过程理论分析学派早在 20 世纪 60 年代就开始研究有限样本情况下的机器学习问题。由于涉及艰涩的数学理论和方法论上的重大革新,90 年代以前并没有提出能够将其理论付诸实践的方法,加之当时正处于神经网络的其他学习方法飞速发展的时期,因此这些研究并没有得到充分的重视。90 年代初期,有限样本的机器学习理论逐渐成熟起来,形成了一个较为完善的理论体系——统计学习理论(statistical learning theory,简称 SLT)^[1~6]。同时神经网络(ANN)等较新兴的启发式机器学习方法的研究则遇到了推广能力差的困难,如神经元网络结构的确定、过学习和欠学习、局部收敛等问题。目前 SLT 与在此基础上发展起来的 SVM 机器学习方法已经成为机器学习领域最新的研究热点。

作为统计学习理论的重要创始人和支持向量机的发明者,V N Vapnik 先后于 1995 年和 1999 年两次出版了该研究领域的重要著作《The Nature of Statistical Theory》,该书重点研究了统计学习理论中的四个方面的重要问题,即经验风险最小化下学习过程一致性的充分必要条件;收敛速度的非渐进理论;学习机推广能力的控制策略以及如何构造理想的学习机。该书为推动统计学习理论和支持向量机的研究起到了十分重要的作用。我国学者张学工将该书译成了中文,为推动 SLT 和 SVM 在国内的研究和应用起到了积极的作用。

第二节 基于 SLT 的机器学习理论的基本观点

一、机器学习问题的本质表示

机器学习的目的可以看作是利用给定的有限数量的训练样本对某系统输入输

出之间依赖关系进行估计,使它能够对未知输出作出尽可能准确的预测。

定义 14.2.1 根据样本学习的机器学习的一般模型 M 是由产生器、训练器、学习机器三个部分组成的三元结构,即 $M = (G, S, LM)$ 。

G 是样本产生器,能从固定但未知的概率分布函数 $F(x)$ 中独立产生随机向量 $x \in R^n$ 。

S 是训练器,对每个输入向量 x 返回一个确定的输出值 y ,产生输出的根据是同样固定但未知的条件分布函数 $F(y|x)$ 。根据联合分布

$$F(x, y) = F(x)F(y|x)$$

抽取出的 l 个独立分布观察数据对

$$(x_1, y_1), \dots, (x_l, y_l)$$

构成训练集。

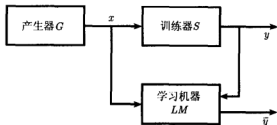


图 14-1 从样本学习的一般模型

LM 是学习机器,能在一组函数集 $f(x, \alpha), \alpha \in \Lambda$ (其中 Λ 是参数集合) 中选择出能够使输出 \bar{y} 最好地逼近训练响应 y 的函数 $f(x, \alpha_0)$ 。各组成的结构关系如图 14-1 所示。训练之后的学习机器必须对任意输入 x 给出输出 \bar{y} 。使该函数最小化风险泛函 $R(\alpha)$:

$$R(\alpha) = \int L(y, f(x, \alpha)) dF(x, y), \quad (14.2.1)$$

其中, $f(x, \alpha), \alpha \in \Lambda$ (其中 Λ 是参数集合) 称学习函数集或预测函数集, α 为函数的广义参数。 $L(y, f(x, \alpha))$ 为在给定输入 x 下训练器输出 y 与学习器给出的 $f(x, \alpha)$ 之间损失的期望。通过选择不同形式的损失函数可以构成不同类型的学习机器。

例如: 如果选择

$$L(y, f(x, \alpha)) = \begin{cases} 0, & \text{如果 } y = f(x, \alpha), \\ 1, & \text{如果 } y \neq f(x, \alpha), \end{cases}$$

且 $y \in \{0, 1\}, f(x, \alpha) \in \{0, 1\}$, 则构成模式识别问题。选择 $(y - f(x, \alpha))^2$ 为损失函数, $y, f(x, \alpha) \in R$ 则成为函数回归估计问题。令 $p(x, \alpha)$ 为密度函数集, 若

$L(p(x, \alpha)) = -\log p(x, \alpha)$ 则构成概率密度估计问题。上述三种问题成为最主要的三大类学习问题。

更一般地, 学习问题可以这样定义:

定义 14.2.2 设有定义在空间 Z 上的概率测度 $F(z)$, 考虑函数的集合 $Q(z, \alpha)$, $\alpha \in \Lambda$, 学习的目标是极小化风险泛函

$$R(\alpha) = \int Q(z, \alpha) dF(z), \quad \alpha \in \Lambda, \quad (14.2.2)$$

其中概率测度 $F(z)$ 未知, 但给定了独立同分布样本

$$z_1, z_2, \dots, z_l, \quad (14.2.3)$$

z 代表了数据对 (x, y) , $Q(z, \alpha)$ 就是待定的损失函数。

对上述学习问题, 有很多学习机可以完成 (如神经网络等), 那么怎样才算一个好的学习机器呢? 一般而言, 主要从 3 个方面评价一个学习机^[2]:

- (1) 学习机器的通用性如何?
- (2) 学习过程是否能以较快的速度收敛?
- (3) 推广能力如何?

关于通用性和收敛性, 很多文献都有所讨论和定义, 这里重点给出推广能力的概念。

定义 14.2.3^[7] 学习机的推广能力是指对一个经过训练并成功收敛于 $Q(z, \alpha_0)$, $\alpha_0 \in \Lambda$ 的学习机 LM_0 , 从概率测度 $F(z)$ 中随机产生样本 z' , $Q(z', \alpha_0)$ 趋近于零的概率 $P(LM_0)$ 。

二、经验风险最小化原则

定义 14.2.2 给出的一般学习问题的学习目标在于使期望风险 $R(\alpha)$ 达到最小化, 但是, 由于对待学习问题已知的全部信息只有样本式 (14.2.3), 式 (14.2.2) 的期望风险并无法计算。因此传统的学习方法中采用了所谓经验风险最小化 (ERM) 的原则, 即用样本定义经验风险:

$$R_{\text{emp}}(\alpha) = \frac{1}{l} \sum_{i=1}^l Q(z_i, \alpha). \quad (14.2.4)$$

可见在回归估计中广为采用的最小二乘法、密度估计的最大似然法只不过是 ERM 原则的一个特例。对模式识别损失函数, 经验风险就是训练样本错误率; 对函数逼近的损失函数, 经验风险就是最小二乘法; 而采用概率密度估计的损失函数的 ERM 原则就等价于最大似然方法。

事实上, 用 ERM 准则代替期望风险最小化是以样本足够大为假设前提的, 在样本数量有限情况下, 是没有充分的理论证据成立的. 但这种思想却在多年的机器学习方法研究中占据了主要地位. 人们将大部分注意力集中到如何更好地最小化经验风险上, 而实际上, 很多问题中的样本数目不可能是无穷大, 因此在有限样本下 ERM 准则得到的结果并不一定能保证使真实风险也较小.

定义 14.2.4 设 $Q(z, \alpha_l)$ 是对给定的独立同分布观测 z_1, \dots, z_l 使经验风险泛函

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^l Q(z_i, \alpha)$$

最小化的函数. 如果下面两个序列概率收敛于同一个极限, 即

$$R(\alpha_l) \xrightarrow{P} \inf_{\alpha \in \Lambda} R(\alpha), \quad R_{emp}(\alpha_l) \xrightarrow{P} \inf_{\alpha \in \Lambda} R(\alpha), \quad (14.2.5)$$

则称 ERM 原则对函数集 $Q(z, \alpha), \alpha \in \Lambda$ 和概率分布函数 $F(z)$ 是一致的, 如图 14-2 所示.

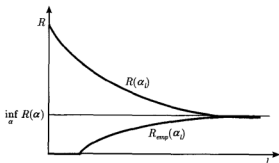


图 14-2 期望风险和经验风险都收敛于最小可能风险的情况

对函数集 $Q(z, \alpha), \alpha \in \Lambda$, 定义其子集 $\Lambda(c)$ 如下:

$$\Lambda(c) = \left\{ \alpha : \int Q(z, \alpha) dF(z) > c, \alpha \in \Lambda \right\}. \quad (14.2.6)$$

如果对函数集的任意非空子集 $\Lambda(c), c \in (-\infty, +\infty)$ 都有

$$\inf_{\alpha \in \Lambda(c)} R_{emp}(\alpha) \xrightarrow{P} \inf_{\alpha \in \Lambda(c)} R(\alpha) \quad (14.2.7)$$

成立, 则 ERM 方法对函数集 $Q(z, \alpha), \alpha \in \Lambda$ 和概率分布函数 $F(z)$ 是非平凡一致的.

也就是说, 一个 ERM 方法, 如果把函数集中取得风险最小值的函数去掉后 (即去掉平凡一致性的情况) 仍然能够满足式 (14.2.7) 的收敛条件, 则这个 ERM 方法是收敛的.

Vapnik 和 Chervonenkis 于 1989 年提出了 ERM 方法非平凡一致收敛的充要条件, 被称为学习理论的关键定理^[2,8].

定理 14.2.1 对函数集 $Q(z, \alpha), \alpha \in \Lambda$, 满足条件:

$$A \leq \int Q(z, \alpha) dF(z) \leq B \quad (A \leq R(\alpha) \leq B),$$

那么, ERM 方法非平凡一致收敛的充要条件是: 经验风险 $R_{emp}(\alpha)$ 在函数集 $Q(z, \alpha), \alpha \in \Lambda$ 上在如下意义下一致收敛于实际风险 $R(\alpha)$:

$$\lim_{l \rightarrow \infty} P \left\{ \sup_{\alpha \in \Lambda} (R(\alpha) - R_{emp}(\alpha)) > \varepsilon \right\} = 0, \forall \varepsilon > 0, \quad (14.2.8)$$

称这种一致收敛为单边收敛. 相对于一致双边收敛的情况,

$$\lim_{l \rightarrow \infty} P \left\{ \sup_{\alpha \in \Lambda} |R(\alpha) - R_{emp}(\alpha)| > \varepsilon \right\} = 0, \forall \varepsilon > 0, \quad (14.2.9)$$

可参阅文献^[1~3]了解关于一致双边收敛的 VC 熵的充要条件.

在探讨结构风险最小化原则之前, 需要用到一个核心概念 VC 维.

三、VC 维

为了研究学习过程一致收敛的速度和推广性, 统计学习理论定义了一系列有关函数集学习性能的指标, 其中最重要的是 VC 维 (Vapnik-Chervonenkis Dimension). 下面给出 VC 维的直观定义^[2,3].

定义 14.2.5 对一个指示函数集 $Q(z, \alpha), \alpha \in \Lambda$, 能够被集合中的函数以所有可能的 2^h 种方式分成两类的向量 z_1, \dots, z_h 的最大数目 h (也就是能够被这个函数集打散的最大样本数目) 称为 $Q(z, \alpha), \alpha \in \Lambda$ 的 VC 维. 若对任意的样本数目 n , 总存在一个 n 个向量的集合可以被函数集打散, 则函数集的 VC 维是无穷大. 若 $A \leq Q(z, \alpha) \leq B, \alpha \in \Lambda$ 是一个以常数 A 和 B 为界的实函数集合, 则考虑其指示器集合 $I(z, \alpha, \beta) = \theta\{Q(z, \alpha) - \beta\}, \alpha \in \Lambda, \beta \in (A, B)$, 其中, $\theta(z)$ 是阶跃函数. 则实函数集 $Q(z, \alpha), \alpha \in \Lambda$ 的 VC 维就是其相应的指示器集合 $I(z, \alpha, \beta), \alpha \in \Lambda, \beta \in (A, B)$ 的 VC 维.

VC 维反映了函数集的学习能力, 直接影响着学习机器的推广性能. VC 维越大则学习机器越复杂 (容量越大). 遗憾的是, 目前尚没有通用的关于任意函数集 VC 维计算的理论, 只对一些特殊的函数集知道其 VC 维. 例如平面中直线的 VC 维是 3 (见图 14-3). n 维坐标空间中的线性函数的集合的 VC 维是 $n+1$. 对于一些比较复杂的学习机器 (如 ANN), 其 VC 维除了与函数集 (神经网络结构) 有关外, 还受学习算法等的影响, 其确定更加困难. 对于给定的学习函数集, 如何 (用理论或实验的方法) 计算其 VC 维是当前统计学习理论中有待研究的一个问题^[9].

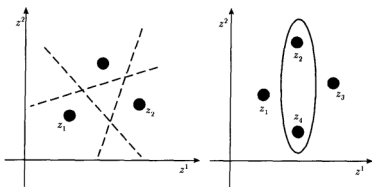


图 14-3 平面中直线的 VC 维是 3 的举例

四、推广性的界理论

对于一个学习机，我们关心的是该学习机对超出训练样本点之外样本数据的处理能力，即学习机的推广性。统计学习理论系统地研究了对于各种类型的函数集，其经验风险和实际风险之间的关系，即推广性的界。分三种情况讨论。

(一) 完全有界函数集

设 $A \leq Q(z, \alpha) \leq B, \alpha \in \Lambda$ 是完全有界函数的集合，那么，经验风险 $R_{emp}(\alpha)$ 和实际风险 $R(\alpha)$ 之间以至少 $1 - \eta$ 的概率满足如下关系^[10]：

$$\begin{aligned} R(\alpha) &\leq R_{emp}(\alpha) + \frac{B-A}{2}\sqrt{\delta}, \\ R(\alpha) &\geq R_{emp}(\alpha) - \frac{B-A}{2}\sqrt{\delta}. \end{aligned} \quad (14.2.10)$$

下面不等式以至少 $1 - 2\eta$ 的概率对使经验风险最小的函数 $Q(z, \alpha_l)$ 成立：

$$R(\alpha_l) - \inf_{\alpha \in \Lambda} R(\alpha) \leq (B-A)\sqrt{-\ln \eta / 2l} + \frac{B-A}{2}\sqrt{\delta}. \quad (14.2.11)$$

(二) 完全有界非负函数集

设 $0 \leq Q(z, \alpha) \leq B, \alpha \in \Lambda$ 是完全有界非负函数的集合，那么，下面的不等式以至少 $1 - \eta$ 的概率同时对 $0 \leq Q(z, \alpha) \leq B, \alpha \in \Lambda$ 的所有函数满足如下关系^[10]：

$$R(\alpha) \leq R_{emp}(\alpha) + (B\delta/2) \left[1 + \sqrt{1 + 4R_{emp}(\alpha)/B\delta} \right]. \quad (14.2.12)$$

下面不等式以至少 $1 - 2\eta$ 的概率对使经验风险最小的函数 $Q(z, \alpha_l)$ 成立：

$$R(\alpha_l) - \inf_{\alpha \in \Lambda} R(\alpha) \leq B\sqrt{-\ln \eta / 2l} + B\delta/2 \left[1 + \sqrt{1 + 4/\delta} \right]. \quad (14.2.13)$$

(三) 无界非负函数集

设 $0 \leq Q(z, \alpha), \alpha \in \Lambda$ 是无界非负函数的集合, 如果不提供关于函数集和 / 或概率测度的额外信息, 是不可能得到描述学习机器推广能力的不等式的. 由于这种情况比较复杂, 可参阅 Vapnik 在 1979 年和 1998 年的有关结论^[1].

在上述讨论中, 如果函数集包含无限多个元素且 VC 维 h 是有限的, 则

$$\delta = 4 \frac{h(\ln(2l/h) + 1) - \ln(\eta/4)}{l}. \quad (14.2.14)$$

如果 $Q(z, \alpha), \alpha \in \Lambda$, 包含有限数目的 N 个元素, 则

$$\delta = 2 \frac{\ln N - \ln \eta}{l}, \quad (14.2.15)$$

这里, l 为样本数目.

对上式, 当 l/h 较大时, δ 就较小, 式 (4.14) 就变得较小, 于是实际风险就接近经验风险的取值. 而当 l/h 较小时, 经验风险 $R_{emp}(\alpha)$ 并不能保证小的实际风险 $R(\alpha)$.

这一关于推广能力界的结论从理论上说明了根据有限已知样本进行一般问题估计的实际风险值, 不仅取决于经验风险最小化, 而且取决于由待估计问题的 VC 维决定的置信范围的大小, 可直观表示为

$$R(\alpha) \leq R_{emp}(\alpha) + \phi(h, l), \quad (14.2.16)$$

$\phi(h, l)$ 表示置信范围, h 为函数的 VC 维, l 为样本大小.

置信范围和学习机器的 VC 维 h 及训练样本数 l 有关. 在有限训练样本下, 学习机器的 VC 维越高 (复杂性越高), 则置信范围越大, 导致真实风险与经验风险之间可能的差别越大. Vapnik 指出^[1], 寻找更好地反映学习机器能力的参数和得到更紧的界是学习理论今后的研究方向之一.

五、结构风险最小化原则

根据前文对 ERM 原则及其推广能力的有关结论, 机器学习过程不但要使经验风险最小, 还要使 VC 维尽量小, 以缩小置信范围, 才能取得较小的实际风险, 即对未来样本有较好的推广性. ERM 原则在样本有限时是不合理的. 如果一个复杂的机器, 其置信范围很大, 即使可以把经验风险最小化为零, 在测试集上的错误数目仍可能很大. 这叫过学习现象. 为避免过学习, 必须构造 VC 维小的学习机器. 但另一方面如果函数集的 VC 维小, 那么就难以逼近训练数据. 因此这是一对矛盾. 在构造学习机时, 可以采用两种策略: 即神经网络采用的策略和支持向量机采用的策略.

(1) ANN 采用了保持置信范围 (通过选择一个适当构造的机器) 并最小化经验风险的策略. 然而, 这个机器如何构造才能使置信范围最小, ANN 并没有明确的依据.

(2) 而 SVM 采用的是保持经验风险固定 (例如等于零) 并最小化置信范围。

显然从获得良好推广能力的角度, SVM 比 ANN 要高明得多。

统计学习理论提出的新的策略^[6]是把函数集构造为一个函数子集序列,使各个子集按照 VC 维的大小排列;在每个子集中寻找最小经验风险,在子集间折衷考虑经验风险和置信范围,从而取得实际风险的最小,如图 14-4 所示。这种思想称作结构风险最小化 (Structural Risk Minimization) 原则,简称 SRM 原则。

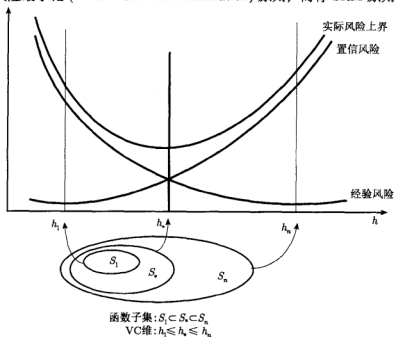


图 14-4 结构风险最小化示意图

统计学习理论还给出了合理的函数子集结构应满足的条件以及在 SRM 原则下实际风险收敛的性质。实现 SRM 原则可以有两种思路。一是在每个子集中求最小经验风险,然后选择使最小经验风险和置信范围之和最小的子集。显然这种方法比较费时,当子集数目很大甚至是无穷多时是不可行的。因此有第二种思路,即设计函数集的某种结构使每个子集中都能取得最小的经验风险 (如使训练误差为 0),然后只需选择适当的子集使置信范围最小,则这个子集中使经验风险最小的函数就是最优函数。支持向量机方法实际上就是这种思想的具体实现。

第三节 支持向量机算法

支持向量机 (Support Vector Machines) 简称 SVM, 是 Vapnik 等根据统计学习理论中的结构风险最小化原则提出的。SVM 能够尽量提高学习机的推广能力,

即使由有限训练样本得到的决策规则对独立的测试集仍能够得到较小的误差。此外,支持向量机算法是一个凸二次优化问题,能够保证找到的极值解就是全局最优解。这些特点使支持向量机成为一种优秀的学习算法。SVM 是统计学习理论中最年轻的内容,也是最实用的部分。其核心内容是在 1992 到 1995 年间提出的^[11],目前仍处在不断发展阶段。

一、广义最优分类超平面

假定训练数据

$$(x_1, y_1), \dots, (x_l, y_l), x \in R^n, y \in \{+1, -1\} \quad (14.3.1)$$

可以被一个超平面

$$(w \cdot x) - b = 0 \quad (14.3.2)$$

分开,如果这个向量被超平面没有错误地分开,并且离超平面最近的向量与超平面之间的距离是最大的,则这个超平面为最优超平面。

为了描述分类超平面,使用以下的形式:

$$\begin{cases} (w \cdot x_i) - b \geq 1, & \text{if } y_i = 1, \\ (w \cdot x_i) - b \leq -1, & \text{if } y_i = -1. \end{cases}$$

采用这些不等式的一种紧凑形式:

$$y_i[(w \cdot x_i) - b] \geq 1, \quad i = 1, \dots, l. \quad (14.3.3)$$

容易验证,最优超平面就是满足条件式 (14.3.3) 并且使得

$$\phi(w) = \|w\|^2 \quad (14.3.4)$$

最小化的超平面。

对一个超平面

$$(w^* \cdot x) - b = 0, \quad \|w^*\| = 1, \quad (14.3.5)$$

如果它以如下的形式将向量 x 分类:

$$y = \begin{cases} 1, & \text{若 } (w^* \cdot x) - b \geq \Delta, \\ -1, & \text{若 } (w^* \cdot x) - b \leq -\Delta, \end{cases} \quad (14.3.6)$$

则称之为 Δ 间隔分类超平面,并有以下关于 Δ 间隔分类超平面集合的 VC 维的定理。

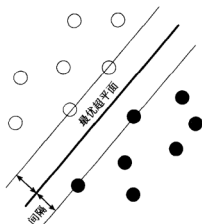


图 14-5 线性可分情况下的最优分类面

定理 14.3.1^[1,2] 设向量 $x \in X$ 是一个半径为 R 的球中的一个向量, 那么 Δ 间隔分类超平面集合的 VC 维 h 以下面的不等式为界:

$$h \leq \min \left(\left\lceil \frac{R^2}{\Delta^2} \right\rceil, n \right) + 1. \quad (14.3.7)$$

二、支持向量机分类算法的推导

要构造最优超平面, 必须用系数的模最小的超平面把属于两个不同类的样本集中的向量 x_i 分开. 为此需求解下面的二次规划问题: 最小化泛函

$$\phi(w) = \frac{1}{2}(w \cdot w), \quad (14.3.8)$$

约束条件为不等式类型:

$$y_i[(w \cdot x_i) - b] \geq 1, \quad i = 1, 2, \dots, l. \quad (14.3.9)$$

这个优化问题的解是由下面的拉格朗日泛函的鞍点给出的:

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^l \alpha_i \{[(x_i \cdot w) - b]y_i - 1\}, \quad (14.3.10)$$

其中, α_i 为拉格朗日乘子. 对该拉格朗日函数关于 w, b 求其最小值, 和关于 $\alpha_i > 0$, 求其最大值.

在鞍点上, w_0, b_0 和 α^0 必须满足以下条件:

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial b} = 0, \quad \frac{\partial L(w_0, b_0, \alpha^0)}{\partial w} = 0. \quad (14.3.11)$$

即最优超平面有以下特性:

1) 系数 α_i^0 必须满足约束

$$\sum_{i=1}^l \alpha_i^0 y_i = 0, \quad \alpha_i^0 \geq 0, \quad i = 1, 2, \dots, l; \quad (14.3.12)$$

2) 最优超平面 (向量 w_0) 是训练集中的向量的线性组合:

$$w_0 = \sum_{i=1}^l y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0, \quad i = 1, 2, \dots, l; \quad (14.3.13)$$

3) 根据 Kuhn-Tucker 条件^[1], 只有支持向量才在 w_0 的展开中具有非零的系数 α_i^0 , 支持向量为使得不等式 (14.3.9) 等式成立的向量, 有

$$w_0 = \sum_{\text{支持向量}} y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0. \quad (14.3.14)$$

又根据 Kuhn-Tucker 条件, 最优超平面的充分必要条件是分类超平面满足条件:

$$\alpha_i^0 \{[(x_i \cdot w_0) - b_0] y_i - 1\} = 0, \quad i = 1, 2, \dots, l. \quad (14.3.15)$$

把 w_0 的表达式代入拉格朗日函数中, 并考虑 Kuhn-Tucker 条件, 问题转化为对偶问题, 即在非负象限

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, l, \quad (14.3.16)$$

并且

$$\sum_{i=1}^l \alpha_i y_i = 0 \quad (14.3.17)$$

下最大化泛函:

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j (x_i \cdot x_j). \quad (14.3.18)$$

这是一个简单的二次规划问题, 文献 [12] 提供了求解这一二次型问题的特殊方法.

如果数据为线性不可分的情况, 构造最优超平面, 只需要增加一个松弛项 $\xi_i \geq 0$, 将约束条件变为

$$y_i [(w \cdot x_i) - b] \geq 1 - \xi_i, \quad i = 1, 2, \dots, l. \quad (14.3.19)$$

将目标函数改为最小化

$$\phi(w, \xi) = \frac{1}{2} (w \cdot w) + C \sum_{i=1}^l \xi_i, \quad (14.3.20)$$

即折中考虑最少错分样本和最大分类间隔, 其中 $C > 0$ 是一个常数, 它控制对错分样本的惩罚程度. 广义最优分类面的对偶问题与线性可分情况下几乎完全相同, 只是条件式 (14.3.16) 变为

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, l. \quad (14.3.21)$$

支持向量机实现了下述思想^[5]: 通过某种事先选择好的非线性映射将输入向量 x 映射到一个高维特征空间 Z , 在这个空间中构造最优分类超平面. 对于 N 维空间中的线性函数, 其 VC 维为 $N+1$, 但根据定理 14.3.1, 即使在十分高维的空间中也可以得到较小 VC 维的函数集, 以保证有较好的推广性. 同时, 通过把原问题转化为对偶问题, 计算的复杂度不再取决于空间维数, 而是取决于样本数, 尤其是样本中的支持向量数. 这些特点使有效地对付高维问题成为可能.

对非线性问题, 可以通过非线性变换转化为某个高维空间中的线性问题, 在变换空间求最优分类面.

1992 年, Boser, Guyon and Vapnik^[13] 发现, 为了在特征空间 Z 中构造最优分类超平面, 并不需要以显式形式来考虑特征空间.

考虑 Hilbert 空间中内积的一个一般表达:

$$(z_i \cdot z) = K(x, x_i), \quad (14.3.22)$$

其中 z 是输入空间中的向量 x 在特征空间中的像. $K(x, x_i)$ 可以是满足 Mercer 定理 [1] 的任意对称函数.

这样以来, 支持向量机的决策函数可表示为

$$f(x) = \text{sgn} \left(\sum_{\text{支持向量}} y_i \alpha_i K(x_i, x) - b \right), \quad (14.3.23)$$

它等价于在高维特征空间 $\psi_1(x), \dots, \psi_N(x)$ 中的线性决策函数. 上式要求得系数 α_i , 只要寻找泛函

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (14.3.24)$$

的最大值, 约束条件仍然为

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, l, \quad \sum_{i=1}^l \alpha_i y_i = 0. \quad (14.3.25)$$

选择不同的函数作为内积的回旋, 可以实现输入空间中不同类型的非线性决策面的学习机器. 常用的学习机器有以下三种:

1) 多项式学习机器, 核函数为

$$K(x, x_i) = [(x \cdot x_i) + 1]^d; \quad (14.3.26)$$

2) 径向基函数机器, 核函数为

$$K(x, x_i) = K_r(\|x - x_i\|); \quad (14.3.27)$$

3) 两层神经网络, 核函数为

$$K(x, x_i) = \text{Sigmoid}[v(x \cdot x_i) + c]. \quad (14.3.28)$$

SVM 分类函数形式上类似于一个神经网络, 输出是中间节点的线性组合, 每个中间节点对应一个支持向量, 如图 14-6 所示。

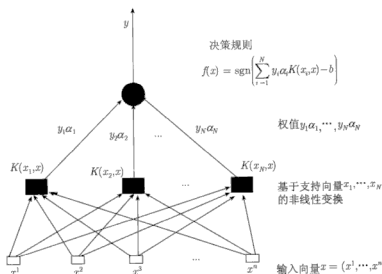


图 14-6 支持向量机的实现形式

第四节 算 例

为可视化方便, 这里以 2 维数据的分类为例, 给出了一个 SVM 分类的算例。训练数据由 2 类共 34 个 2 维数据组成 (见图 (a))。选取不同的分类核函数其分类结果如下图 14-7 所示。

上图中, 圆圈代表第一类数据, 菱形代表第二类数据, “+” 标记为支持向量, 实曲线为最优分类面, 两侧虚线为支持向量确定的分类面最大间隔。图 (a) 为原始训练样本的分布, 图 (b)、图 (c) 为选择径向基函数核函数的分类结果, 图 (d) 为多项式核函数的处理结果, 图 (e) 为 Sigmoid 核函数的处理结果。

对应各种核函数所获得的支持向量及其对应的拉格朗日系数等有关实验数据见表 14-1。

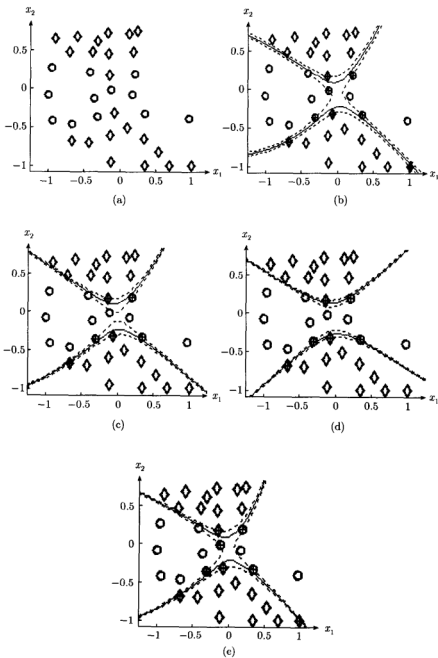


图 14-7 支持向量机分类结果

表 14-1 实验结果

图 b $t = 0.5S$	Rbf 核函数: $K(x, x_i) = \exp(-\ x - x_i\ ^2 / 2\gamma^2), \gamma = 1$								
	支持向量	0.4811	0.4992	0.416	0.4402	0.4463	0.4387	0.5825	0.4372
	(8 个)	0.522	0.4556	0.4516	0.496	0.4577	0.4214	0.371	0.5202
	拉氏系数	173.3	288.7	1143.4	812.5	1504.6	225.6	7.7	693.8
图 c $t = 0.5S$	Rbf 核函数: $K(x, x_i) = \exp(-\ x - x_i\ ^2 / 2\gamma^2), \gamma = 2$								
	支持向量	0.4811	0.4992	0.416	0.4463	0.4387	0.4372		
	(6 个)	0.522	0.4556	0.4516	0.4577	0.4214	0.5202		
	拉氏系数	7857	4939	39457	34514	8163	9636		
图 d $t = 0.5S$	多项式核函数: $K(x, x_i) = [(x \cdot x_i) + 1]^d, d = 2$								
	支持向量	0.4811	0.4992	0.416	0.4463	0.4387	0.4372		
	(6 个)	0.522	0.4556	0.4516	0.4577	0.4214	0.5202		
	拉氏系数	345.4	183.3	1696.3	1436.8	341.5	444.5		
图 e $t = 0.5S$	Sigmoid 核函数: $K(x, x_i) = \tanh[3\ x - x_i\ / 34 + 2]$								
	支持向量	0.4811	0.4992	0.416	0.4402	0.4463	0.4387	0.5825	0.4372
	(8 个)	0.522	0.4556	0.4516	0.496	0.4577	0.4214	0.371	0.5202
	拉氏系数	34.41	39.1	187.753	116.633	236.754	29.006	0.101	111.37

第五节 本章小结

支持向量机是一种适用于小样本情况的基于统计学习理论的机器学习方法。本章深入分析了 SVM 的统计学理论基础, 阐述了从经验风险最小化到结构风险最小化的统计学习变革历程, 推导并分析了一个 SVM 学习机算法。最后利用一个可视化方便的 2 维两类分类的实例给出了在以往向基、多项式和 Sigmoid 函数为核函数情况下的分类结果。

参 考 文 献

- [1] Vapnik V. Statistical Learning Theory. N.Y.: Springer, 1998
- [2] 张学工 (译). 统计学习理论的本质. 北京: 清华大学出版社, 2000. 226
- [3] Vapnik V. The Nature of Statistical Learning Theory. N.Y.: Springer, 1995. 188
- [4] Steve R Gunn. Support Vector Machines for Classification and Regression. University of Southampton, 1998
- [5] 张学工. 统计学习理论与支持向量机. 自动化学报, 2000, 26(1): 32~41
- [6] Christopher J C Burges. A tutorial on Support Vector Machines for Pattern Recognition. Bell Lab, Lucent Technologies, Boston: Kluwer Academic Publishers, 1998: 1~43
- [7] 马笑潇. 智能故障诊断中的机器学习新理论及其应用. 重庆: 重庆大学, 2002
- [8] Vapnik V N and Chervonenkis A Ja. The Necessary and Sufficient Conditions for Consistency of the Method of Empirical Risk Minimization. Pattern Recogn. and Image Analysis, 1991, 1(3): 284~305
- [9] Vapnik V, Levin E, Le Cun Y. Measuring the VC-dimension of a Learning Machine. Neural

- Computation, 1994(6): 851~876
- [10] Burges C J C. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 1998, 2(2):121~167
- [11] Cortes C and Vapnik V. Support vector networks. Machine Learning, 1995, 20: 273~297
- [12] More J J and Toraldo G. On the Solution of Large Quadratic Programming Problems with Bound Constraints. SLAM Optimization, 1991, 1(1): 93~113
- [13] Boser B, Guyon I and Vapnik V N. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the fifth annual workshop on Computational learning theory. New York: ACM Press, 1992: 4~152

第十五章 基于 SVM 的多类分类算法 及其在故障诊断中的应用

第一节 引言

SVM 是一种崭新的机器学习方法,在理论上具有很突出的优势,但与其理论研究相比,应用研究则相对比较滞后,目前只有较有限的实验研究报道^[1,2]. SVM 的应用应该是一个大有作为的方向.目前,在模式识别方面最突出的应用研究是贝尔实验室对美国邮政手写数字库进行的实验^[3],用 SVM 方法得到的识别结果明显优于决策树和多层神经网络.相关的应用还包括 SVM 与神经网络相结合对笔迹进行在线适应^[4],MIT 用 SVM 进行的人脸检测实验也取得了较好的效果,可以较好地学会在图像中找出可能的人脸位置^[5].其他有报道的实验领域还包括文本识别、人脸识别、三维物体识别、遥感图像分析等^[3].可见, SVM 的应用大都集中于图像处理领域,文献^[6]给出了将 SVM 机器学习方法引入工业领域进行故障诊断的试验,利用基于 SVM 的二叉树多类分类方法有效地解决了柴油机振动信号的故障识别问题.

由于 SVM 算法是从 2 类分类问题推导出来的,在解决像故障诊断等典型的多类分类问题时会遇到困难.将 SVM 机器学习方法延伸到多类分类问题目前还处于初步研究阶段^[7],已经提出的解决这个问题的思路主要有 2 大类^[8]:

(1) 构造多个 2 类分类器组合起来完成多类分类,该思路源于传统的模式识别方法解决多类问题.根据训练样本组成的不同,又分成“one against one”和“one against all”两种类型^[9,10],前者的每一个 SVM 的训练样本是由两个不同类别的样本组成,需要构造 $(k(k-1)/2)$ 个 SVM 才能完成整个分类任务, k 为类别数;后者的每个 SVM 的训练样本都由全部样本组成,需构造 k 个支持向量机才能完成.这类思路简单有效,但存在无法识别的阴影区域,而且重复训练的样本较多.

(2) 只使用一个 SVM 机实现多个分类输出 (“all together”)^[11,12],这种思路涉及十分复杂的优化问题,对训练样本数目相对较大时,需要很长的运算时间^[9],而且分类精度不很理想.

第二节 基于二叉树的多级 SVM 分类器

文献^[6]提出了一种基于 SVM 的二叉树多类分类算法,并将其用于多类故障

诊断,取得了较好的效果.

该方法是一种适合故障诊断等多类决策的二元数分类方法.属于第一大类,但采用了一种“one against others”的分类器构造策略.方法具有算法简单、直观,重复训练样本少的优点.下面给出算法说明.

给定一个 k 类分类问题,学习样本为 $(x_1, y_1), \dots, (x_l, y_l), x_j \in R^n, j = 1, \dots, l, y_j \in \{1, \dots, k\}$. 该分类算法是一个四元组:

$$\langle F, P, \text{SVM}, CS \rangle. \quad (15.2.1)$$

$F = \{f_1, \dots, f_i, \dots, f_k\}$, 是二叉树的终止节点集合, 由待识别系统的 k 个可能的模式集构成.

$P = \{p_1, \dots, p_i, \dots, p_k\}$ 表示各模式类发生的优先级, 由各种状态发生的频率的高低排序确定. 最可能出现的状态优先级定为 p_1 , 发生可能性最低的状态定为最后一级 p_k . P 决定了分类器中多个支持向量机的安排方式.

$\text{SVM} = \{\text{SVM}_{p_1}, \dots, \text{SVM}_{p_i}, \dots, \text{SVM}_{p_{k-1}}\}$ 是由所设计的 $k-1$ 个支持向量机组成的二叉树的全部非终止节点集合. 对一个 k 类分类问题, 需要构造 $k-1$ 个 SVM 机. 其中第 i 个 SVM 决定的模式类优先级为 p_i .

$SC = \{SC_1, \dots, SC_i, \dots, SC_k\}$ 为属于 k 个模式类别的全部学习样本集合, 其中 $SC_i = \{(x_1, y_i), \dots, (x_j, y_i), \dots, (x_l, y_i)\}$, 表示第 i 类的样本组成, $x_j \in R^n, y_i \in \{1, \dots, k\}, \sum_{SC} l_i = l$, 构成全部学习样本.

第 i 级支持向量 SVM_{p_i} 的训练样本 S_{p_i} 按下述原则确定:

$$\begin{cases} S_{p_1} = \text{全部样本}, \\ S_{p_i} = \overline{SC_{i-1}}, \\ S_{p_i} = \overline{SC_i} \oplus SC_i, \end{cases} \quad i = 1, \dots, k-1, \quad (15.2.2)$$

$\overline{SC_i}$ 表示不属于第 $1, \dots, i$ 类模式的样本全体. 可见随着优先级的降低, 训练样本数逐渐减少. 第 i 个 SVM 解决以下问题^[9]:

$$\min_{\omega_i, b_i, \xi_i} \frac{1}{2} (\omega_i)^T \omega_i + C \sum_{j=1}^l \xi_i^j. \quad (15.2.3)$$

如果, $y_j = i$, 则 $(\omega_i)^T H(x_j) + b_i \geq 1 - \xi_i^j$.

如果, $y_j \neq i$, 则 $(\omega_i)^T H(x_j) + b_i \leq -1 + \xi_i^j$.

$\xi_i^j \geq 0, j = 1, \dots, l, H(x_j)$ 是训练样本 x_j 在高维特征空间中的映射, C 是不可分情况下的惩罚因子, 以降低训练错误数目.

这样可以得到 $k-1$ 个决策函数:

$$\begin{aligned} &(\omega_1)^T H(x) + b_1, \\ &\vdots \\ &(\omega_{k-1})^T H(x) + b_{k-1}. \end{aligned} \quad (15.2.4)$$

对每一级 SVM 训练后找出对应该级的支持向量, 建立最优分类超平面. 由于 $k-1$ 个 SVM 是按照优先级由高到低排列的, 新模式产生时, 只需要按照二叉树由高到低进行搜索, 就可以得出结论.

可见, 该多类分类方法总体上属于基于 SVM 的多类分类策略中的第一大类, 对于多类多峰分布问题, 通过优先级的确定可以迅速建立一个简单的二叉树分类器, 它不是企图用一种算法、一个决策规则去把多个类别一次分开, 而是采用分级的形式, 把一个复杂的多类别分类问题转化为若干个简单的 2 类分类问题, 使分类问题逐步得到解决. 该方法不存在无法分类的阴影问题, 因此分类准确度较高. 相对于 “one against all” 和 “one against one” 策略, 大大降低了样本的重复训练量. 下面给出证明.

对于一个由 l 个训练样本组成的 k 个模式的分类问题, 上述方法的重复训练样本量是最小的.

\mathcal{R}_{others} 定义为重复训练样本数与重复次数的乘积, 则

$$\mathcal{R}_{others} = \sum_{i=2}^{k-1} l_i \cdot (i-1) + l_k \cdot (k-2), k \geq 3. \quad (15.2.5)$$

证明 第 1 类样本只训练 1 次, 故重复训练量 $r_{other}^1 = 0$, 如果 $i \neq k$, 则第 i 类样本参与训练 i 次, 故重复训练量 $r_{other}^i = l_i \cdot (i-1)$; 如果 $i = k$, 则第 i 类样本参与训练的次数等于第 $i-1$ 类的训练次数, 故重复训练量 $r_{other}^k = l_k \cdot (k-2)$. 因此, 全部 k 类样本的重复训练量求和后为表达式 (15.2.5).

“one against all” 方法, 需要构造 k 个支持向量机, 重复训练样本量为

$$\mathcal{R}_{all} = \sum_{i=1}^k l_i \cdot k - \sum_{i=1}^k l_i, k \geq 3. \quad (15.2.6)$$

上式意义比较明确, 前一项为全部 k 类样本总的训练量, 后一项为训练一次的训练量, 二者相减即为重复量.

“one against one” 方法需构造 $k \cdot (k-1)/2$ 个支持向量机, 其重复训练量为

$$\mathcal{R}_{one} = \sum_{i=1}^k l_i \cdot (k-2), k \geq 3 \quad (15.2.7)$$

由于每一类都需要与其他 $(k-1)$ 类参与训练 1 次, 故重复训练量为上式.

例如：令 $l_i = 10, i = 1, \dots, k, k = 5$, 则

$$\mathcal{R}_{others} = \sum_{i=2}^{5-1} 10 \cdot (i-1) + 10 \cdot (5-2) = 90,$$

$$\mathcal{R}_{all} = \sum_{i=1}^5 10 \cdot (5-1) = 50 \times 4 = 200,$$

$$\mathcal{R}_{one} = \sum_{i=1}^5 10 \cdot (5-2) = 150.$$

可见，二叉树的训练样本重复量是最小的，只是“one against all”方法的 45%，和“one against one”方法的 60%，而且，当样本量增大时，其训练样本重复量与其他两种方法的差别会拉大，这大大降低了样本训练量。在目前 SVM 训练时间还较长的现状下，该方法是一种不错的选择。

在应用当中，首先设计第一个 SVM，称为故障检测 SVM，完成系统正常样本与全部故障样本的分类，即检测出系统是否发生故障。如果辨识出系统正常则无须进入下一级 SVM 检验。第二个 SVM 只在故障样本中进行，完成第一主要故障与其他故障类型的分类，往下依次类推，直到第 $k-1$ 个 SVM 输出分类结果为止。这种基于二叉数的按优先级高低排列的多类分类器设计方法具有概念简单、直观、效率高的特点，而且利用了故障类型的先验知识，是一种非常适合故障诊断的多类 SVM 机器学习方法。

算法完成故障诊断的结构如图 15-1 所示。

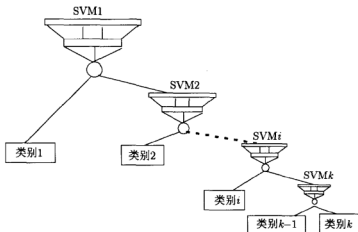


图 15-1 基于 SVM 的多类故障诊断结构

第三节 SVM 用于故障诊断的一般步骤

智能故障诊断面临的一个难题之一是故障特征知识的发现问题，在要求不解

体、实时诊断的情况下, 获取故障的大量有效样例更为困难。而常规的诊断方法大都依赖于大样本情况下的统计特性 (包括神经网络), 当训练样本有限时, 难以保证有较好的分类推广性。因此选择一种具有良好推广特性、适合小样本情况的学习机器进行样本训练是非常关键的。支持向量机满足结构化风险最小化原则, 通过将特征空间映射为 Hilbert 高维空间, 从而有效地降低待求解问题的 VC 维, 达到同时最小化经验风险和置信范围的目的, 是一种可行的选择。

支持向量机应用于故障诊断最大的优势在于它适合于小样本决策。算法分为两个阶段: 首先, 通过已知的正常和各种故障状态下的特征样本对 SVM 进行训练, 找到样本中的支持向量, 据此确定最优分类超平面。然后, 测试集样本根据最优分类面作出分类决策。下面给出 SVM 应用于故障诊断中的具体实现方法。

(一) 学习阶段

Step 1 根据专家经验或动态聚类结果建立训练样本集 $\{x_i, y_i\}, y_i \in \{1, -1\}$ 。

Step 2 选择合适的核函数 $K(x, x_i)$ 及有关参数, 作为高维特征空间在低维输入空间的一个等效形式。选择的依据是 Mercer 定理。核函数通常选择多项式函数、径向基函数或 Sigmoid 函数等。

Step 3 输入样本正规化。目的是为了将输入数据标定在 Kernel 核函数要求的范围之内, 例如, 对于多项式、径向基、Sigmoid 函数将 x_i 规定在 $[-1, 1]$ 内。

Step 4 构造 Kernel 矩阵 $H(l, l)$ 。

Step 5 在约束条件 $\sum_{i=1}^l \alpha_i y_i = 0, 0 \leq \alpha_i \leq C$ 下, 最大化

$$Q(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (15.3.1)$$

以求解拉格朗日系数 α 。这是通过将输入空间映射为高维特征空间, 然后在低维输入空间运用高维特征空间的内积的等效形式——Kernel 核函数实现的。已经提出了几种快速算法^[13,14]来解决这个问题。 C 值的选择可以控制是完全可分 ($C = \infty$) 情况或不可分情况下的推广特性。

Step 6 找出支持向量 sv , 求解分类超平面系数 b 。

Step 7 建立训练数据的最优决策超平面, 完成训练过程。

(二) 新故障模式的识别阶段

Step 1 装入 SVM 学习阶段的有关数据 (包括训练数据 $\{x_i, y_i\}$, 系数 α 和 b 以及支持向量 sv)。

Step 2 根据式 $f(x') = \sum_{\text{支持向量}} y_i \alpha_i K(x_i, x') - b$, 计算新输入特征数据 x' 决策输出值。

Step 3 利用指示函数将 $f(x')$ 归为 $\{-1, 1\}$, 作出分类决策。

SVM 机器学习方法的本质在于能够在有限特征信息情况下, 最大限度地发掘数据中隐含的分类知识。在训练阶段, 特征数据所隐含的各种故障的特征知识是隐式地表达在样本中有限数目的支持向量及其相应的拉格朗日系数 α 上的。也就是说, 除了支持向量之外的其他样本本质上是冗余的。SVM 最重要的特性在于其良好的外推能力, 这一点对于故障诊断而言是有很强的实际意义的。其推广能力是通过由支持向量确定的最优分类超平面的求取实现的, SVM 机器学习方法很好地执行了结构风险最小化原则, 这一统计学习理论的新原则。

由于 SVM 算法是从 2 类分类问题推导出来的, 在解决故障诊断这种典型的多类分类问题时必须解决多类分类的问题。这里采用前文提出的二叉树多类分类方法, 实现了柴油机缸盖振动信号的故障诊断问题。

第四节 基于 SVM 的柴油机故障诊断

柴油机具有振源多、运动部件多、长期工作在高温、高压环境下的特点, 一旦发生故障很不容易诊断。利用缸盖的振动信号特征进行故障诊断是一种经常采用的有效方法^[15]。

这里采用的柴油机缸盖振动信号故障诊断的策略是: 首先对测取的汽缸振动信号, 进行小波包分解, 提取振动信号在不同频带的正交特征参量, 建立学习样本。然后训练由 4 个 SVM 机按照故障优先级不同构成的二叉树多级分类器, 进行故障的检测和辨识。

实验对象为 6135 型柴油机, 通过对其缸盖振动信号进行同步采样, 获取一个完整工作循环内的离散振动信号序列。采样时, 以第一缸燃烧上止点信号作为触发采样信号, 信号长度为每循环 2000 个点, 采样频率为 25kHz。对在不同工作状态下测取的振动信号, 采用 3 阶 Daubechies 小波, 对信号进行 3 层分解, 分解后得到的 8 个频带的宽度均为 1562.5Hz。图 15-2 为新气阀轻微漏气故障时缸盖振动信号的分解波形, 按图号顺序从 (a) 到 (g) 频段由低到高分布。

信号经过小波包分解后, 可以直观地了解故障的特征频带, 但如直接送至学习机进行训练, 则数据量太大。为了降低分类器的复杂度, 可以求出该频带内的幅值谱。然后计算各频带的相对幅值谱值, 作为该频带的特征参量, 就可以大大简化特征空间。由于样本数目较多, 表 15-1 为部分样本特征参量示例。

对上述高维特征数据采用前文论述的多类分类方法进行故障模式决策。6135 型柴油机的气门故障主要有 4 种, 按照故障发生的可能性进行优先级排序分别为排气门间隙过小为第一故障, 排气门间隙过大 (第二故障), 新气阀轻微漏气 (第三故障) 和气阀严重漏气 (第四故障)。在缸盖上测取了大量不同气门间隙状态下的振动信号, 经小波包分解后提取特征向量, 构成 8 维特征参量的输入样本。按照优先

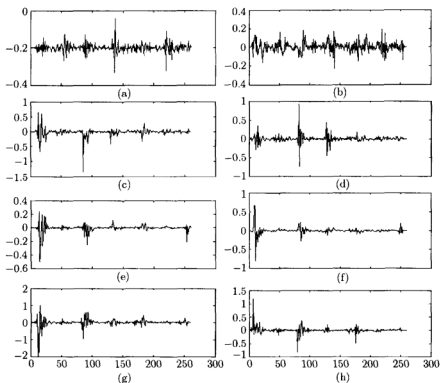


图 15-2 新气阀轻微漏气时的振动信号小波包分解结果

表 15-1 小波包特征参量

相对值	频带 1	频带 2	频带 3	频带 4	频带 5	频带 6	频带 7	频带 8
样本 1	0.07	0.0938	0.0997	0.1288	0.2276	0.1509	0.1184	0.1108
样本 2	0.0919	0.0924	0.1294	0.1019	0.1702	0.1614	0.1242	0.1285
样本 3	0.0711	0.0822	0.202	0.1327	0.041	0.0619	0.3133	0.0959
样本 4	0.0743	0.1192	0.0802	0.1354	0.1958	0.1544	0.1175	0.1232
样本 5	0.0665	0.0803	0.2457	0.1169	0.055	0.0524	0.2969	0.0863
样本 6	0.0676	0.0821	0.1309	0.1254	0.1226	0.1373	0.1676	0.1663
样本 7	0.0698	0.0708	0.1615	0.1048	0.0551	0.0936	0.2967	0.1477
样本 8	0.0674	0.0939	0.1361	0.1266	0.2073	0.1438	0.1164	0.1085
样本 9	0.0725	0.0884	0.1251	0.1436	0.1113	0.1077	0.2201	0.1313
样本 10	0.0694	0.063	0.1632	0.1216	0.0714	0.092	0.2748	0.1446

级由高到低分别训练 4 个支持向量机。实验在主频 700MHz 的 Legend 计算机上进行, 这里给出部分实验结果。

(一) 第一级 SVM

样本构成: 正常及各种故障状态下的振动特征数据, 共 35 组。

取径向基核函数: $K(x, x_i) = \exp(-\|x - x_i\|^2 / 2\gamma^2)$, 表 15-2 是在不同 γ 取值下系统执行时间和训练获得的支持向量的数目. $C = +\infty$.

表 15-2 取径向基核函数的第一级 SVM 训练结果

γ 取值	0.2	0.6	1.0	1.3	1.5	2.0	2.8	4.0	6.0
SV 个数	35	17	7	5	4	3	3	3	3
SV 百分比 (%)	100	48.6	20.0	14.3	11.4	8.6	8.6	8.6	8.6
时间 (秒)	0.88	0.74	0.74	0.78	0.79	0.80	0.73	0.74	0.70

当 γ 取值大于 2.0 以后, 支持向量数目稳定为 3. $\gamma = 2.0$ 时, 对应的支持向量及其拉氏系数如下表:

表 15-3 $\gamma = 2.0$ 时, 求得的支持向量

Sv1	0.0919	0.0924	0.1294	0.1019	0.1702	0.1614	0.1242	0.1285
Sv2	0.0828	0.0909	0.109	0.1291	0.1983	0.1365	0.1394	0.1141
Sv3	0.0676	0.0821	0.1309	0.1254	0.1226	0.1373	0.1676	0.1663

表 15-4 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	5	正常	10.1167
Sv2	7	正常	1.2572
Sv2	16	第三故障	11.3295

取多项式核函数: $K(x, x_i) = [(x \cdot x_i) + 1]^d$, $C = +\infty$, 表 15-5 是在不同 d 取值下系统执行时间和训练获得的支持向量数目.

表 15-5 取多项式核函数的第一级 SVM 训练结果

多项式阶次 d	2	3	4	5	6	7	8	9
SV 个数	3	3	6	5	5	5	3	1
SV 百分比 (%)	8.6	8.6	17.1	14.3	11.4	14.3	8.6	2.9
时间 (秒)	0.77	0.90	0.83	0.78	0.80	0.76	0.84	0.88

$d = 3$ 时获取的支持向量与 $\gamma = 2.0$ 时, Rbf 核函数所得的支持向量完全相同. 拉氏系数如下表:

表 15-6 $d = 3$ 时, 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	5	正常	0.0516
Sv2	7	正常	0.0042
Sv2	16	第三故障	0.0826

取 Sigmoid 核函数: $K(x, x_i) = \tanh(-p1 \cdot (x \cdot x_i)/l + p2)$, $C = +\infty$, 这里 $l = 35$. 以下是在不同参数取值下获得的实验结果.

表 15-7 取 Sigmoid 核函数的第一级 SVM 训练结果

p1, p2 取值	1, 1	1, 2	2, 1	2, 2	2, 3	3, 1	3, 2	3, 3	4, 1
SV 个数	3	2	3	2	2	3	3	3	3
SV 百分比 (%)	8.6	5.7	8.6	5.7	11.4	8.6	8.6	8.6	8.6
时间 (秒)	0.91	0.88	1.04	0.90	0.91	0.90	0.88	0.89	0.90

SV 个数为 3 时, 支持向量与其他核函数的结果相同, 当 $p1 = 2, p2 = 2$ 时, 支持向量数目为 2. 去掉了第 7 号样本.

 表 15-8 $p1 = 2, p2 = 2$ 时, 求得的支持向量

Sv1	0.0919	0.0924	0.1294	0.1019	0.1702	0.1614	0.1242	0.1285
Sv3	0.0676	0.0821	0.1309	0.1254	0.1226	0.1373	0.1676	0.1663

表 15-9 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	5	正常	702.8986
Sv2	16	第三故障	702.8986

分析以上实验结果, 不难发现, 不同核函数对支持向量的影响并不明显, 只是多项式核函数的阶次变动, 对支持向量的个数影响较大. 多层感知器核函数对参数的变化并不明显. 在实际应用中取 RBF 核函数, $\gamma = 2.0$, $C = +\infty$. 对应支持向量如表 15-2.

(二) 第二级 SVM2

样本构成: 气门间隙过小作为第一故障与其他故障状态下的振动特征数据进行分类, 共 28 组. 省略参数选择过程, 直接给出最优参数下的训练结果.

取 RBF 核函数: $C = +\infty$, $\gamma = 2.0$, CPU 运行时间为 0.63 秒.

表 15-10 取径向基核函数时第二级 SVM 的支持向量

Sv1	0.0701	0.0754	0.1831	0.1201	0.0659	0.0586	0.3174	0.1094
Sv2	0.0718	0.0647	0.1587	0.115	0.0706	0.1111	0.2684	0.1397
Sv3	0.0702	0.0642	0.1514	0.1184	0.0618	0.0759	0.2817	0.1762

表 15-11 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	2+7	第一故障	9.5772
Sv2	15+7	第三故障	5.3768
Sv3	18+7	第三故障	4.3271

(三) 第三级 SVM3

样本构成：气门间隙过大故障与其他故障状态下的振动特征数据，共 21 组。省略参数选择过程，直接给出最优参数下的训练结果。

取多项式核函数： $C = +\infty$, $d = 2$ 时，CPU 运行时间为 0.45 秒，获取的支持向量与相应拉氏系数如表 15-12。

表 15-12 取多项式核函数的第三级 SVM 训练结果

Sv1	0.0676	0.0821	0.1309	0.1254	0.1226	0.1373	0.1676	0.1663
Sv2	0.0709	0.0795	0.0999	0.1149	0.094	0.1369	0.2179	0.1859
Sv3	0.0739	0.0834	0.1477	0.12	0.0859	0.114	0.185	0.1901
Sv4	0.0717	0.0867	0.1127	0.1514	0.0929	0.0991	0.2069	0.1786
Sv5	0.0725	0.0884	0.1251	0.1436	0.1113	0.1077	0.2201	0.1313

表 15-13 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	2+14	第二故障	0.4039
Sv2	3+14	第二故障	0.3473
Sv3	15+14	第四故障	0.5577
Sv4	18+14	第四故障	0.1395
Sv5	19+14	第四故障	0.2405

(四) 第四级 SVM4

样本构成：新气阀轻微漏气故障与严重漏气状态下的振动特征数据，共 14 组。省略参数选择过程，直接给出最优参数下的训练结果。

取多项式核函数： $C = +\infty$, $d = 3$ 时获取的支持向量与相应拉氏系数如表 15-14。

表 15-14 取多项式核函数的第四级 SVM 的支持向量

Sv1	0.0718	0.0647	0.1587	0.115	0.0706	0.1111	0.2684	0.1397
Sv2	0.0691	0.0654	0.2199	0.0964	0.069	0.1107	0.2094	0.1601
Sv3	0.0682	0.0675	0.2133	0.1158	0.0616	0.0654	0.2098	0.1985
Sv4	0.0739	0.0834	0.1477	0.12	0.0859	0.114	0.185	0.1901
Sv5	0.0744	0.0885	0.1698	0.1648	0.0647	0.0895	0.185	0.1636
Sv6	0.0725	0.0884	0.1251	0.1436	0.1113	0.1077	0.2201	0.1313

表 15-15 支持向量对应的拉氏系数

SV 向量	样本序号	所属类别	拉氏系数
Sv1	1+14	第三故障	0.0142
Sv2	5+14	第三故障	0.0030
Sv3	6+14	第三故障	0.0089
Sv4	8+14	第四故障	0.0248
Sv5	9+14	第四故障	0.0136
Sv6	12+14	第四故障	0.0175

至此,四级分类支持向量机全部学习完毕,学习机获取了振动特征数据隐含的分类知识。在训练数据集之外,每一类中另抽取 10 组共 5×10 组特征数据来测试机器推广能力。测试实验分两个方面进行,首先测试已经学习过的全部样本,观察只依赖于有限数目的支持向量的记忆能力;然后用各种没有学习过的样本测试学习机的推广能力。

(一) 记忆能力测试

以全部 35 组数据作为输入,送入分类器,结果全部正确,证明了多类分类算法具有记忆功能。

(二) 推广能力测试

以下是从中选取的部分特征数据测试样本。

表 15-16 部分测试样本

正常	0.0674	0.0939	0.1361	0.1266	0.2073	0.1438	0.1164	0.1085
正常	0.0877	0.0768	0.1299	0.1227	0.207	0.1457	0.1279	0.1021
第一故障	0.0684	0.0727	0.2028	0.1181	0.0588	0.0594	0.2928	0.127
第一故障	0.0711	0.0822	0.202	0.1327	0.041	0.0619	0.3133	0.0959
第二故障	0.0699	0.0848	0.1006	0.1183	0.1254	0.1578	0.1441	0.1991
第二故障	0.0704	0.0836	0.1362	0.1211	0.0929	0.1396	0.1612	0.1949
第三故障	0.0694	0.063	0.1632	0.1216	0.0714	0.092	0.2748	0.1446
第三故障	0.0698	0.0708	0.1615	0.1048	0.0551	0.0936	0.2967	0.1477
第四故障	0.0736	0.083	0.1307	0.1559	0.0555	0.0969	0.2041	0.2003
第四故障	0.0834	0.0789	0.138	0.1595	0.0689	0.0937	0.209	0.1686

测试结果见表 15-17。

表 15-17 算法推广能力测试结果

	SVM1	SVM2	SVM3	SVM4
分类样本总数	50	40	30	20
错分数	0	0	1	0
准确率	100%	100%	3.33%	100%

在第二级分类错误的样本为: $x = [0.0704 \ 0.0836 \ 0.1362 \ 0.1211 \ 0.0929 \ 0.1396 \ 0.1612 \ 0.1949]$, 经分析, 发现该类样本具有较高的离散度, 由于训练样本有限, 造成所选支持向量推广能力发生畸变。事实上, x 经决策函数的实际输出为 -0.0301 , 处于决策边界上。这种一般可以通过采用软决策实现正确分类。

图 15-3 所示为第二故障与其他故障 SVM 分类时 2 维可视情况下的分类效果图。图中, (a) 为以第 1, 2 维特征数据进行分类的训练结果; (b) 为以第 2, 3 维特征数据进行分类的训练结果; (c) 为以第 3, 4 维特征数据进行分类的训练结果; (d) 为以第 4, 5 维特征数据进行分类的训练结果; (e) 为以第 5, 6 维特征数据进行分类的训练结果; (f) 为以第 7, 8 维特征数据进行分类的训练结果。

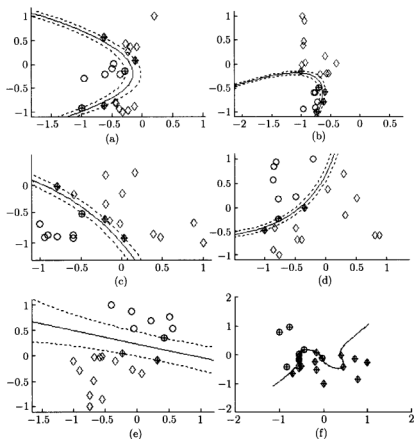


图 15-3 SVM 故障分类 2 维可视效果图

(三) 参数分析

支持向量机中有关参数的选择仍然是一个有待研究的问题^[16]。上一节已经对核函数有关参数通过实验的方法给出了大致分析, 本节主要对惩罚系数进行讨论。

图 15-4 中, 由 (a) 到 (d), C 值依次增大, 图 15-3(f) 是在 C 取无穷大时得到的结果。由此可见: 惩罚因子 C 起到控制学习误差的作用, C 越小, 学习误差越大, 训练得到的 SVM 就有越强的推广能力; C 越大, 学习误差越小, 训练得到的 SVM 的外推能力就弱。可见一个好的学习机器, 是在寻求训练误差和推广能力的一个最佳结合点。

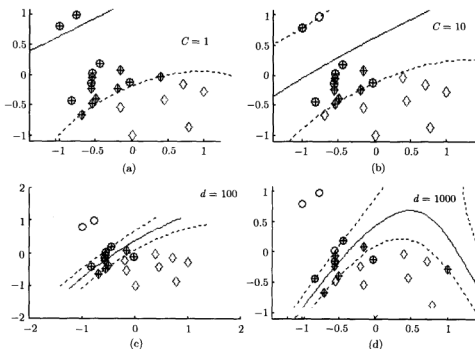


图 15-4 惩罚因子 C 参数的敏感性分析

再看一下 SVM 学习机的收敛速度问题。

共对 234 组 8 维故障特征数据进行训练, 在 CPU 主频为 700MHz 的计算机上进行了测试, SVM 机学习时间如表 15-18 所示。

表 15-18 不同训练集下的 CPU 运行时间比较

N	5	15	25	35	45	60	80	100	120	140	160	180
$T(S)$	0.08	0.14	0.26	0.4	0.62	1.02	1.75	2.85	4.38	6.43	8.60	11.69

运行时间随样本数目增加的趋势图见图 15-4。对于大量样本情况下, 如何提高 SVM 训练速度、改进现有学习算法仍是一个需要解决的问题。不过, 同一般的学习机器 (如 ANN) 比较来看, SVM 仍具有较高的效率而且其收敛过程更容易控制。

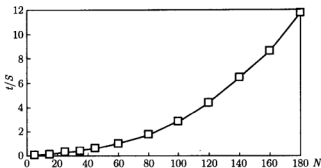


图 15-5 运行时间随样本数目增加的趋势图

第五节 本章小结

SVM 机器学习方法是一种新颖的机器学习方法，在理论上具有严谨的数学基础，应该有着广阔的应用前景，然而实际的应用却非常滞后。由于常规 SVM 算法是从 2 类分类问题推导出来的，在解决类似故障诊断这种典型的多类分类问题存在困难，本章研究了基于 SVM 的多类模式识别算法，探讨了支持向量机用于故障诊断的关键问题。最后以柴油机振动信号的故障诊断为例，利用基于 SVM 的二叉树多类分类方法，成功地实现了柴油机故障检测和故障原因判断，并通过大量的实验，分析了 SVM 理论中有关参数的特性和选择依据。

参 考 文 献

- [1] 张学工. 统计学习理论与支持向量机. 自动化学报, 2000, 26(1):32~41
- [2] Cholkopf B, Sung K-K, Burges C et al. Comparing Support Vector Machines with Gaussian Kernels to Radial basis Function Classifiers. IEEE Trans. on Signal Processing, 1997, 45(11): 2758~2765
- [3] 张学工 (译). 统计学习理论的本质. 清华大学出版社, 2000. 226
- [4] Matic N, Guyon T, Denker J et al. Writer Adaptation for on-line handwritten Character Recognition. In: 2nd Int. Conf. on Pattern Recognition and Document Analysis, 1993:187~191
- [5] Osuna E, Freund R, Girosi F. Training Support Vector Machines: an Application to Face Detection. In: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97). Washington, DC: IEEE Computer Society, 1997. 130
- [6] 马英潇, 黄席樾等. 基于 SVM 的二叉树多类分类算法及其在故障诊断中的应用. 控制与决策, 2003, 18(3):272~276
- [7] Krebel U. Pairwise Classification and Support Vector Machines. In Scholkopf B, editor. Advances in Kernel Methods-Support Vector Learning, MA, 1999: 255~268
- [8] 马英潇. 智能故障诊断中的机器学习新理论及其应用. 重庆: 重庆大学, 2002
- [9] Chih-wei Hsu, Chih-Jen Lin. A Comparison of Methods for Multi-classification Support Vector Machines. [http:// www. csie. nt u. edu. tw/ ~cjlin/bsvm/](http://www.csie.ntu.edu.tw/~cjlin/bsvm/), 2001

-
- [10] Bottou L, Cortes C, et al. Comparison of Classifier Methods: a Case Study in Handwriting Digit Recognition. In Int. Conf. on Pattern Recognition, IEEE Computer Society Press, 1994. 77~87
 - [11] Krebel U. Pairwise Classification and Support Vector Machines. In Scholkopf B, editor. Advances in Kernel Methods-Support Vector Learning, MA, 1999:255~268
 - [12] Weston J and Watkins C. Multi-class Support Vector Machines. In Verleysen M, editor. Proceedings of ESANN99, BrusselsD. Facto Press, 1999
 - [13] Platt J C. Fast Training of SVMs Using Sequential Minimal Optimization. In Scholkopf B, Burges C J C and Smola A J, editors. Advances in Kernel Methods- Support Vector Learning, MIT Press, 1998:185~208
 - [14] Osuna E, Freund R, and Girosi F (1997). An Improved Training Algorithm for Support Vector Machines. In Principe J, Gile L, Morgan N, and Wilson E(Eds.). Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop, New York, 1997: 276~285
 - [15] 耿尊敏, 宋孔杰, 李兆前, 等. 关于柴油机振声特点及动态诊断方法的研究与讨论. 内燃机学报, 1995, 13(2): 140~147
 - [16] Vapnik V. Statistical Learning Theory. N.Y.: Springer. 1998

第十六章 基于支持向量机的函数回归的方法

支持向量机在模式识别领域的应用,实际上是对指示函数进行估计,如果推广到估计实函数,就成为回归问题.通过引入新的损失函数,运用支持向量机学习方法可以实现具有较强鲁棒性的回归,而且回归估计是稀疏的.解的稀疏性对在高维空间中用大量资料估计依赖性关系是非常重要的^[1].

第一节 常用的损失函数的定义

进行函数的估计,需要选择一种损失函数,常用的损失函数有^[4]

(一) 二次损失函数

$$L(y, f(x, \alpha)) = (y - f(x, \alpha))^2.$$

这是在正态加性噪声下以 ERM 原则对一个回归函数的最佳无偏估计.形如图 16-1(a).

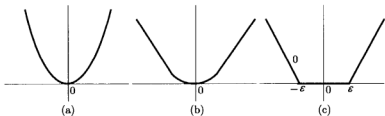


图 16-1 常用的几种损失函数

(二) Huber 损失函数

$$L(y, f(x, \alpha)) = \begin{cases} c|y - f(x, \alpha)| - \frac{c^2}{2}, & \text{如果 } |y - f(x, \alpha)| > c, \\ \frac{1}{2}|y - f(x, \alpha)|^2, & \text{如果 } |y - f(x, \alpha)| \leq c. \end{cases} \quad (16.1.1)$$

这是 Huber 提出的一种鲁棒回归函数, 当噪声是某种固定的噪声与另一个对称连续密度函数的任意噪声的混合时, 该损失函数依最大最小 (Max-Min) 策略具有最佳回归特性. 形如图 16-1(b).

(三) ϵ - 不敏感损失函数

$$L(y, f(x, \alpha)) = L(|y - f(x, \alpha)|_\epsilon) = \begin{cases} 0, & \text{如果 } |y - f(x, \alpha)| \leq \epsilon, \\ |y - f(x, \alpha)| - \epsilon, & \text{其他.} \end{cases} \quad (16.1.2)$$

这是由 Vapnik 提出的对 Huber 损失函数的一种近似形式. 该函数之所以广为使用, 是因其特有的稀疏性, 通常对 ϵ - 不敏感损失函数得到的解的展开式使用最少的支持向量. 形如图 16-1(c).

第二节 函数回归的 SVM 方法

首先考虑一个线性回归问题:

已知一个训练集: $D = \{(x_1, y_1), \dots, (x_1, y_1), \dots, (x_l, y_l)\}$, $x \in \mathbb{R}^n, y \in \mathbb{R}$ 在线性函数集合

$$f(x, \alpha) = w \cdot x + b \quad (16.2.1)$$

中估计回归函数.

把回归估计的问题定义为对一个损失函数进行风险最小化的问题, 用 SRM 原则进行风险最小化时, 最优的回归函数是通过在一定的约束条件下最小化泛函^[1~3]:

$$\phi(w, \xi^*, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i^* + \xi_i) \quad (16.2.2)$$

实现的, 其中, C 是一个给定的常数, ξ_i^*, ξ_i 是松弛变量, 二者规定了函数输出的上、下界, 并定义 $F(\xi, \xi^*) = \sum_{i=1}^l (\xi_i^* + \xi_i)$, 则寻求经验风险最小化时的线性函数的系数 w 和 b , 是通过最小化 $F(\xi, \xi^*)$ 实现的.

当采用 ϵ - 不敏感损失函数, $L(y, f(x, \alpha)) = L(|y - f(x, \alpha)|_\epsilon)$ 时, 式 (16.2.2) 最小化的约束条件为

$$\begin{aligned} y_i - (w \cdot x_i) - b - \epsilon &\leq \xi_i^*, \quad i = 1, \dots, l, \\ (w \cdot x_i) + b - y_i - \epsilon &\leq \xi_i, \quad i = 1, \dots, l, \\ \xi_i^* &\geq 0, \xi_i \geq 0, i = 1, \dots, l. \end{aligned} \quad (16.2.3)$$

对这样一个二次优化问题, 可以通过最大化以下二次型求解:

$$W(\alpha, \alpha^*) = -\varepsilon \sum_{i=1}^l (\alpha_i^* + \alpha_i) + \sum_{i=1}^l y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i \cdot x_j), \quad (16.2.4)$$

并有约束条件

$$0 \leq \alpha_i^*, \alpha_i \leq C, i = 1, \dots, l, \quad (16.2.5)$$

$$\sum_{i=1}^l \alpha_i^* = \sum_{i=1}^l \alpha_i. \quad (16.2.6)$$

可以求得向量

$$w_0 = \sum_{i=1}^l (\alpha_i^* - \alpha_i) x_i. \quad (16.2.7)$$

根据 Karush-Kuhn-Tucker(KKT) 条件^[1], 有 $\alpha_i^* \times \alpha_i = 0$ 成立, 也就是只有支持向量对应的拉格朗日乘子不等于 0. 因此可以只采用训练样本中的少数支持向量就可以实现函数估计.

对于非线性回归问题, 可以通过非线性变换将输入向量映像到高维特征空间, 转化为类似的线性回归问题加以解决. 为了避免高维特征空间中的“维数灾难问题”, 采用 Hilbert 空间中内积的回旋形式, 用输入空间的一个核函数等效高维特征空间的内积形式.

选择不同的核函数作为内积的回旋, 可以实现输入空间中不同类型的非线性决策面的学习机器. 核函数 $K(x, x_i)$ 选取的要求仍然是必须满足 Mercer 条件, 常用的核函数有:

- (1) 生成多项式的核函数: $K(x, x_i) = [(x \cdot x_i) + 1]^d$;
- (2) 生成径向基函数的核: $K(x, x_i) = K_r(\|x - x_i\|)$;
- (3) 生成两层神经网络的核: $K(x, x_i) = \text{Sigmoid}[v(x \cdot x_i) + c]$;
- (4) 生成 1 阶 B 样条函数的核:

$$K(x, x_i) = 1 + x x_i + \frac{1}{2} |x - x_i| (x \wedge x_i)^2 + (x \wedge x_i)^3 / 3, \quad (16.2.8)$$

其中, $(x \wedge x_i) = \min(x, x_i)$, 这样, 式 (16.2.4) 变为

$$W(\alpha, \alpha^*) = -\varepsilon \sum_{i=1}^l (\alpha_i^* + \alpha_i) + \sum_{i=1}^l y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j). \quad (16.2.9)$$

约束条件与式 (16.2.5) 和式 (16.2.6) 相同. 这样, 回归函数变为

$$f(x) = \sum_{\text{支持向量}} \beta_i K(x, x_i) + b, \quad (16.2.10)$$

$$\beta_i = \alpha_i^* - \alpha_i, \quad i = 1, \dots, l,$$

通过控制 C 和 ε 两个参数, 可以控制支持向量机的推广能力.

选择其他的损失函数的回归函数的估计的推导过程, 基本相同, 不再赘述.

不同的实值函数估计问题需要不同的逼近函数集, 对于回归问题, 构造反映逼近函数特性的核函数是十分重要的. 下面给出构造用于逼近定义在向量空间 $X \subset R^n$ 上的多维函数的核的方法, 向量 $x = (x^1, \dots, x^n)$ 的所有坐标定义在相同的有限或无限区间 I 上.

现在假设对任意坐标 x^k 都给出了完备正交基 $b_{ik}(x^k)$, $i = 1, 2, \dots$. 考虑 n 维空间中下面的基函数集合:

$$b_{i_1, i_2, \dots, i_n}(x^1, \dots, x^n) = b_{i_1}(x^1) b_{i_2}(x^2) \cdots b_{i_n}(x^n). \quad (16.2.11)$$

这些函数是由每个坐标的基函数通过它们的直积 (张量积) 构造的, 函数集 (16.2.11) 是一个 $X \subset R^n$ 中完备正交基.

对多维核函数的构造有以下重要定理 [1,3]:

定理 16.2.1 设一个多维函数集合是由作为单位坐标基函数之张量积的基函数定义的, 那么这个 n 维基上的内积的核是一维核的积, 即

$$K(x, x_i) = \prod_{k=1}^n K(x^k, x_i^k). \quad (16.2.12)$$

此式表明, n 维核函数可以由 n 个一维核构成.

第三节 基于 SVM 的故障趋势预测研究

相对于在模式识别领域的应用试验, SVM 在函数回归方面的应用研究更少.

Vapnic 给出了一个对由 506 个 13 维数据进行的试验, 通过采用 4 阶多项式核函数, 试验得出的测试实际值与预测值的均方误差明显低于 Bagging 和 AdaBoost 算法的试验结果 [2]. 显示了 SVM 在回归估计方面的优势. 文献 [5] 将 SVM 回归方法引入工业故障预测领域, 实现了工业过程参数的预测. 本章将支持向量机的回归算法, 引入故障诊断领域, 提出了基于 SVM 回归方法的故障趋势预测的方法, 实现了具有较强容错特性的趋势预测.

利用 SVM 回归方法进行特征参数的跟踪和预测的步骤如下:

Step 1 确定特征参数 x 的历史数据: 最近 n 次诊断时间 t_i 和对应的特征参数值 x_i , $i = 1, 2, \dots, n$.

Step 2 回归问题定义为 $x = f(t, \alpha)$, $\alpha \in \Lambda$, 表示广义函数参数.

Step 3 构造用 SVM 方法求解回归问题的核函数 $K(t, t_i)$.

Step 4 采用支持向量方法, 用核函数求解回归问题, 找到历史数据中的支持向量 t_i , $i = 1, \dots, n$ 和相应拉氏系数 β_i , $i = 1, \dots, n$.

Step 5 确定回归输出 $f(t) = \sum_{\text{支持向量}} \beta_i K(t, t_i) + b$.

Step 6 做出故障趋势预测决策输出.

这里以“Tennessee Eastman”工厂的仿真数据^[6~9]为例, 运用上面提出的步骤, 进行以时间为自变量的故障趋势预测的仿真研究.

选择信号波动随机性较强的反应器液位(图 16-2)作为故障监测的特征变量. 这样, 可以测试趋势预测的鲁棒性.

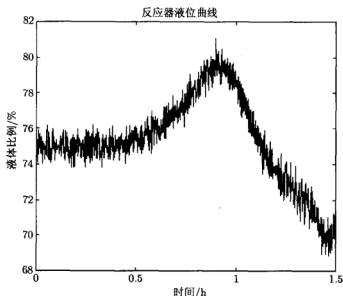


图 16-2 反应器液位输出曲线

当全部 12 个操作变量的输入为表 16-1 所示时, 反应器液位会经过缓慢振荡后超出警戒范围, 并同时导致出现严重的反应器过压故障.

表 16-1 导致反应器过压故障的操作变量输入

编号	1	2	3	4	5	6
名称	D 进料	E 进料	A 进料	A+C 进料	回收阀开度	净化阀开度
变量相对值 (%)	63.0530	53.9800	24.6440	61.3020	22.2100	40.0640
编号	7	8	9	10	11	12
名称	分离器阀开度	解析器阀开度	蒸汽阀开度	反应器冷却水量	冷凝器冷却水量	风扇速度
变量相对值 (%)	38.1000	46.5340	47.4460	41.1060	18.1130	51.2000

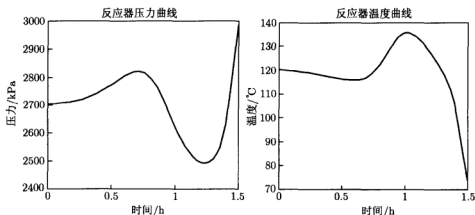


图 16-3 对应的反应器压力和温度变化曲线

原始测量数据的采样周期 5.4 秒,为了验证在 ϵ -不敏感损失函数下,基于 SVM 回归的稀疏性,分别以 20 倍和 40 倍采样周期取观测数据,分别获得 50 组、25 组稀疏数据。以下是回归结果。

1. 20 倍采样周期取观测数据,取 1 阶 B 样条核函数 (式 16.2.10),并以前 30 组数据作为训练输入,估计全部 50 组数据。

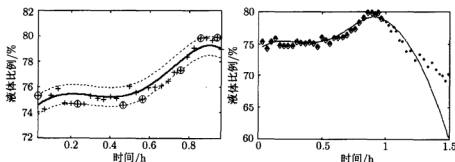
定义已训练样本的测试平均误差 $e_{trn} = \left(\sum_{i=1}^l |Y_{trn_i} - Y_{tst_i}| \right) / l$, l 为训练样本数目, Y_{trn_i}, Y_{tst_i} 分别为第 i 个训练样本的实际值和测试输出值。

定义测试样本的测试平均误差 $e_{tst} = \left(\sum_{i=1}^l |Y_i - Y_{tst_i}| \right) / l$, l 为训练样本数目, Y_i, Y_{tst_i} 分别为第 i 个测试样本的实际值和测试值。SVM 机的有关参数和运算结果见表 16-2。

表 16-2 参数选择和实验结果

参数选择				回归结果			
C	样条阶次	ϵ	SV 的数目	CPU 时间	e_{trn}	e_{tst}	
$+\infty$	1	0.75	7(23.3%)	2.66 秒	0.4645(%)	1.3188(%)	
Sv 编号	Sv1	Sv2	Sv3	Sv4	Sv5	Sv6	Sv7
样本号	1	7	14	17	23	26	29
拉氏系数	28.32	-2.84	-6.49	-12.99	-21.33	26.11	22.09

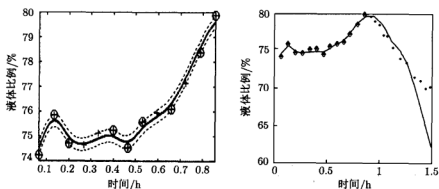
图 16-4 给出了图示结果。其中,左图为训练结果,红圈内“+”标志表示支持向量,两条虚线围成训练误差 ϵ 管道,黑实线为回归逼近结果,“+”标志为 ϵ 管道内的训练样本;右图为预测结果,菱形内“.”标志为训练样本的实际值,“.”为测试样本的实际值,黑线为估计输出结果(以下实验结果,采用相同标志,不再说明)。

图 16-4 $C = +\infty$, $\varepsilon = 0.75$, 对 30 个样本训练和回归结果

2. 40 倍采样周期选取观测数据, 并以前 15 组数据作为训练输入, 采用 1 阶 B 样条核函数估计全部 25 组数据. 实验结果见表 16-3. 图 16-5 给出了图示结果.

表 16-3 参数选择和实验结果

参数选择				回归结果					
	C	样条阶次	ε	SV 的数目	CPU 时间	e_{trn}		e_{test}	
	$+\infty$	1	0.25	9(69.2%)	0.60 秒	0.2199		1.0132	
Sv 编号	Sv1	Sv2	Sv3	Sv4	Sv5	Sv6	Sv7	Sv7	Sv8
样本号	1	2	3	6	7	8	10	12	13
拉氏系数	-206.25	591.78	-430.11	282.06	-420.31	259.24	-106.31	-21.92	83.77

图 16-5 $C = +\infty$, $\varepsilon = 0.25$, 对 15 个样本训练和回归结果

上面给出的两个实验, 分别是根据液位的前 30 个和前 15 个测量数据预测后 20 个和后 10 个数据的液位输出. 而且第 2 个实验的数据更为稀疏. 可见, SVM 方法可以实现故障趋势预测. 在当前时刻 t , 根据前 n 次测量数据, 即可以判断 t 时刻之后的输出, 如果预测输出可能超过规定值 (出现故障), 就可以在当前时刻, 建议采取措施, 避免潜在故障的产生.

3. 有关参数分析

用 SVM 方法进行趋势预测,重要的是选择 ε -不敏感损失函数的 ε 宽度,该参数是控制支持向量数目的最关键参数.但是如何选择 ε 宽度,至今仍是一个没有解决的问题,文献 [10~12] 提出了几种选择 ε 的方法.但是总起来讲,经验仍占重要成分. ε 的直观解释如下 [1]:

以精度 ε 逼近函数 $f(x)$,也就是用另一个函数 $f(x, \alpha_0)$ 来描述函数 $f(x)$,要构造这样一个函数,取一个弹性 ε 管道,并把 $f(x)$ 放到这个 ε 管道中,因为弹性管道总趋向于变平,因此,它会碰到函数 $f(x)$ 的一些点,于是这个管道的轴线就定义了 $f(x)$ 的 ε 逼近 $f(x, \alpha_0)$.管道碰到函数 $f(x)$ 上的点的坐标定义了支持向量.核函数 $K(x, x_i)$ 则描述了管道的弹性规律.

为了验证 ε 取值不同对回归结果的影响,笔者进行了多组实验,结果如图 16-6 所示.实验仍然采用 1 阶 B 样条核函数,左图取 ε 为 0.0;中图取 0.75;右图取 1.25.

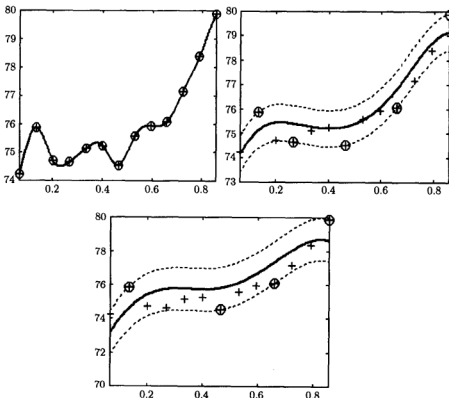


图 16-6 不同 ε 取值下,对 15 个样本的训练结果

不同 ε 取值对支持向量个数的影响,见图 16-7 左曲线.该图反映的大致规律是 ε 取值越大,所求得的支持向量就越少,回归曲线就越平坦.图 16-7 右曲线为训练样本量对训练时间的影响曲线.样本量增加导致训练时间几乎呈指数增加.以

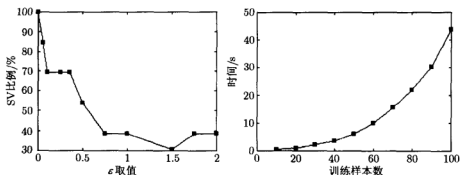


图 16-7 ϵ -支持向量数目关系曲线和训练样本量-CPU 时间关系实验曲线

上结果的实验条件与前相同。

从上述实验结果可以得到以下结论:

- (1) 用 SVM 方法进行函数回归, 即使对在一定噪声影响下的信号仍具有较好的函数估计能力;
- (2) 在可以接受的限度内具有一定的预测外推能力;
- (3) 该方法在对稀疏数据的函数估计上有更高的利用价值;
- (4) 不足之处是, 当训练样本量较大时, 训练时间较长;
- (5) 该方法中 ϵ 参数的取值对预测精度较为敏感。

由此可见, 基于 SVM 机器学习方法的故障趋势预测是一种解决小样本预测的比较有前途的方法。

第四节 本章小结

本章研究了 SVM 回归算法的基本实现方法, 给出了运用 SVM 机器学习方法进行故障趋势预测的方法, 通过对“Tennessee Eastman”工厂的实际数据进行仿真, 结果表明用 SVM 方法进行故障趋势预测, 具有较强的抗噪能力, 在样本量有限情况下, 以 ϵ -不敏感损失函数的回归结果具有良好的解的稀疏性。通过大量实验, 验证了 ϵ 取值不同对估计结果的影响, 分析了 ϵ 取值对支持向量数目的控制作用。

参 考 文 献

- [1] 张学工 (译). 统计学习理论的本质. 北京: 清华大学出版社, 2000. 226
- [2] Vapnik V. Statistical Learning Theory. N.Y.: Springer, 1998
- [3] Vapnik V. The Nature of Statistical Learning Theory. N.Y.: Springer, 1995. 188
- [4] Steve R Gunn. Support Vector Machines for Classification and Regression. University of Southampton, 1998

-
- [5] 马笑潇, 黄席樾等. 基于支持向量机的故障过程趋势预测研究. 系统仿真学报, 2002, 14(11): 1548~1551
 - [6] Downs J J and Vogel E F. A Plant-wide Industrial Control Problem. Comput. Chem. Engng., 1993(17): 245~255
 - [7] Mcavoy T J and Ye N. Base Control for the Tennessee Eastman Problem. Comput. Chem Engng., 1994(18): 383~413
 - [8] Ricker N L. Decentralized Control of the Tennessee Eastman Challenge Process. J. Proc Control, 1996(6): 205~221
 - [9] Ashish Singhal. Tennessee Eastman Plant Simulation with Base Control System of McAvoy and Ye, <http://depts.washington.edu/control/LARRY/TE/download.html>
 - [10] Smola A, Murata N and et al. Asymptotically Optimal Choice of ε -loss for Support Vector Machines. In Niklasson L and et al editors. Proceedings of the 8th Int. Conf. on Artificial Neural Networks, Perspectives in Neural Computing, Berlin, 1998:105~110
 - [11] Scholkopf B, Smola A and Williamson R. A New Parameterization of Support Vector Machines. In EUROCOLT, 1998
 - [12] Scholkopf B, Bartlett P and Smola A and Williamson R. Support Vector Regression with Automatic Accuracy Control. In Nillasson L, et al editors. Proceedings of the 8th Int. Conf. on Artificial Neural Networks, Perspectives in Neural Computing, Berlin, 1998: 111~116

中英文词汇对照

B

背包问题 Knapsack problem

比特分配 Bit allocation

不变区 Constant region

不动点 Fixed point

不动点定理

Banach fixed-point theorem Banach

部分异步并行实现

Partially asynchronous parallel
implementation

B 细胞 B cell

C

超变异 Hypermutation

车间作业调度问题

Job-shop scheduling problem

车辆路径问题 Vehicle routing problem

尺度不变性 Scale invariance

初次免疫应答 Initial immune response

D

带聚类处理的蚁群算法

Clustering processing ant colony
algorithm

迭代函数系统 Iterated function system

度量 Metric

独特型免疫网络

Idiotypic immune network

多功能缩印机

Multiple reduction copying machine

D 块 Domain block

E

二次分配问题

Quadratic assignment problem

F

范数 Norm

仿射映射(变换)

Affine mapping (transformation)

分形 Fractal

分形空间 Fractal space

分形块编码 Fractal block coding

分形图像编码(压缩)

Fractal image coding (compression)

分形维数 Fractal dimension

峰值信噪比 Peak signal-to-noise ratio

G

骨髓 Bone

H

盒维数 Box-counting dimension

混沌游戏算法 Chaos game algorithm

J

激励度 Stimulation levell

基于混合行为的蚁群算法

Hybrid behavior based ant colony
algorithm

基于排序的蚂蚁系统

Rank-based version of ant system

基于图的蚂蚁系统元启发式

Graph-based ant system(GBAS)

渐进解码 Progressive decompression

紧集 Compact set

经验风险最小化 ERM

巨噬细胞 Macrophage cell

均方根 Root mean square

均方误差 Mean square error

K

抗体 Antibody

抗体浓度 Antibody density

抗原 Antigen

抗原呈递细胞 Antigen presenting cell

抗原肽 Antigenic peptide

可变区 Variable region

克隆选择 Clone selection

跨尺度冗余 Across-scale redundancy

K-旅行商问题

K-person traveling salesman problem

L

淋巴细胞 Lymphocyte

率失真曲线 Rate-distortion curve

旅行商问题 Traveling salesman problem

M

蚂蚁系统 Ant system

免疫 Immunity

免疫记忆 Immune memory

免疫算法 Immune algorithm

免疫网络 Immune network

免疫网络算法 Immune network algorithm

免疫细胞 Immune cell

免疫系统 Immune system

免疫遗传算法 Immune genetic algorithm

免疫应答 Immune response

N

内积 Inner product

P

匹配追赶算法 Matching pursuit algorithm

拼贴定理 Collage theorem

Pareto 最优解 Pareto optimal solution

Q

亲和成熟 Somatic maturation

亲和力和 Affinity

R

R 块 Range block

S

时间相关信息素下界的 GBAS

GBAS with time-dependent lower
pheromone bound

时间相关蒸发系数的 GBAS

GBAS with time-dependent evaporation
factor

矢量量化 Vector quantization

失真测度 Distortion measure

随机梯度下降

Stochastic gradient descent

T

特征尺度 Characteristic size

体液免疫 Humoral immune

停滞现象 Stagnation behavior

同步并行实现

Synchronous parallel implementation

统计学习理论

Statistical learning theory

图着色问题 Graph coloring problem

T 细胞 T cell

V

VC 维 Vapnik-chervonenkis dimension

W

外激素 Pheromone

网络路由问题 Network route problem

网络抑制 Network suppression

伪随机比例状态转移规则

Pseudo random proportional

state transition rule

X

细胞免疫 Cell immune

吸引子 Attractor

先天性免疫 Innate immune

相似度 Similarity level

小生境 Niche

虚拟码本 Virtual codebook

Y

压缩变换 Contractive transformation

压缩因子 Contractivity factor

蚁群系统 Ant colony system

蚁群优化 Ant colony optimization

蚁群优化元启发式

Ant colony optimization meta

heuristic

Z

再次免疫应答 Secondary immune response

在线延迟信息素更新

Online delayed pheromone update

在线逐步信息素更新

Online step-by-step pheromone update

增强型学习系统

Reinforcement learning system

支持向量机 Support vector machines

主要组织相容性复合物

Major histocompatibility complex

自催化行为 Autocatalytic behavior

自适应免疫 Adaptive immune

自适应蚁群算法

Adaptive ant colony algorithm

自相似性 Self-similarity

最大-最小蚂蚁系统 Max-min ant system

最终压缩 Eventual contractivity